Nonlinear Finite Elements: Modeling and Simulation of Earthquakes, Soils, Structures and their Interaction

Prof. Dr. Boris Jeremić; Борис Јеремић
University of California, Davis

with significant contributions, as noted in Chapters, by:

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prof. Dr. Zhaohui Yang</td>
<td>University of Alaska, Anchorage</td>
</tr>
<tr>
<td>Dr. Zhao Cheng</td>
<td>Itasca International Inc., Minneapolis</td>
</tr>
<tr>
<td>Dr. Guanzhou Jie</td>
<td>Wells Fargo Securities, New York</td>
</tr>
<tr>
<td>Dr. Nima Tafazzoli</td>
<td>Tetra Tech EBA, Vancouver</td>
</tr>
<tr>
<td>Dr. Matthias Preisig</td>
<td>GeoMod, Lausanne</td>
</tr>
<tr>
<td>Dr. Panagiota Tasiopoulou</td>
<td>Fugro, Athens</td>
</tr>
<tr>
<td>Prof. Dr. Federico Pisanò</td>
<td>Delft University of Technology, Delft</td>
</tr>
<tr>
<td>Prof. Dr. José Abell</td>
<td>University of the Andes, Santiago</td>
</tr>
<tr>
<td>Mr. Kohei Watanabe</td>
<td>Shimizu Corporation, Tokyo</td>
</tr>
<tr>
<td>Dr. Yuan Feng</td>
<td>TuSimple, San Diego</td>
</tr>
<tr>
<td>Dr. Sumeet Kumar Sinha</td>
<td>University of California, Davis</td>
</tr>
<tr>
<td>Ms. Fatemah Behbehani</td>
<td>University of California, San Diego</td>
</tr>
<tr>
<td>Dr. Han Yang</td>
<td>University of California, Davis</td>
</tr>
<tr>
<td>Dr. Hexiang Wang</td>
<td>University of California, Davis</td>
</tr>
</tbody>
</table>

Copyright by Boris Jeremić
Typeset in \TeX

http://sokocalo.engr.ucdavis.edu/~jeremic/LectureNotes/
Motivation

The main motivation for development of the Real-ESSI system, comprised of these lecture notes and accompanying modeling tools, computational libraries and visualization tools, is to help research and teach modeling and simulation for civil engineering mechanics problems. Focus is on development and use of methods that reduce Kolmogorov complexity and modeling uncertainty. In other words, focus is on development and use of methods that predict and inform rather than fit. These lecture notes, in particular, are being developed to document research, teaching and practical problem solving work for Real-ESSI problems (Realistic modeling and simulation of Earthquakes, Soils, Structures and their Interaction). Almost all of the theories, formulations and algorithms described here can be directly analyzed using Real-ESSI Simulator system (http://real-ssi.info/; http://real-ssi.us/) on local computers or on Cloud Computers (Amazon Web Services, AWS, on Marketplace, search for ESSI). A number of theories and formulations, related to Real-ESSI problems, developed by us and others, as referenced, are collected within these Lecture Notes in order to have one location, one write-up, with all/most necessary material for the analysis of ESSI problems. These Lecture Notes are in perpetual development, and chapters and sections are being edited and added as you read this. In that sense, these Lecture Notes are not "polished" and there are some rough edges, however improvement work is underway.

Work on these lecture notes was motivated by a number of books and lecture notes that I have enjoyed over many years, (Bathe, 1982), (Bathe and Wilson, 1976), (Felippa, 1992b, 1989, 1993; Felippa and Park, 1995), (Willam, 1993), (Sture, 1993), (Lubliner, 1990), (Crisfield, 1991), (Chen and Han, 1988a), (Zienkiewicz and Taylor, 1991a,b), (Argyris and Mlejnek, 1991), (Malvern, 1969), (Saouma, 1992-2013), (Dunica and Kolundžija, 1986), (Kojić, 1997), (Hjelmstad, 1997), (Oberkampf et al., 2002). I particularly enjoyed book by Bathe (1982), the only one I had partial access to in the late ’80s, with all the examples that could be worked out on paper. In ’89 I managed to purchase a book by Zienkiewicz (1977) for US$50, from my first salary as a young engineer in Energoprojekt Company, from a colleague. In the early ’90s, I was lucky to get exposed to early, draft versions of books by Kojić (1993); Kojić (1997). Few years later, I enjoyed lectures and lecture notes by Felippa (1992b, 1989, 1993); Felippa and Park (1995), Willam (1993), and Sture (1993).

Current version of lecture notes, the one in front of you, aims to extend concepts described by my Teachers and Professors. Presented formulations and implementations are available within the Real-ESSI Simulator http://real-ssi.info/, http://real-ssi.us/. A number of provided models, some very simple, some more sophisticated, and some very sophisticated, can be analyzed and results visualized using Real-ESSI Simulator on local computers or on Cloud Computers (Amazon Web Services, AWS).
Contributions

Useful contributions were also made by the following students, colleagues and collaborators (other than those listed on the front page): Mr. Babak Kamranimoghadam (بابک کامرانی), Mr. Chang-Gyun Jeong (정창균), Mr. Chao Luo (羅超), Mr. Max Sieber, Mr. Antonio Felipe Salazar, and Mr. Borko Miladinović (Борко Миладиновић).

Comments

Comments, corrections, edits &c. are much appreciated! Special thanks to (in chronological order): Miroslav Živković (Мирослав Живковић), Dmitry J. Nicolsky, Andrzej Niemunis, Robbie Jaeger, Yiorgos Perikleous (Γιώργος Περικλέους), Robert Roche, Viktor Vlaski, Edison Lam.

The best way to send a comment on these lecture notes is by email, however please read the following NOTE about sending an email to me. It would be great if you can place the following in the subject line of your email: Draft CompMech Lecture Notes. This will be much appreciated as it will help me filter your email and place it in Draft CompMech Lecture Notes email-box that I regularly read.

Acknowledgement

Many developments described in these lecture notes, developed over many years, were made possible in collaborators with, and with financial support from the: US-DOE, US-NRC, CNSC/CCSN, US-NSF, CalTrans, CH-ENSI/IFSN, ATC/US-FEMA, UN-IAEA, US-ACE, and NASA. Their support and collaboration is much appreciated!
Software

Theoretical and computational developments described in these lecture notes are implemented in a program Real-ESSI Simulator (http://real-essi.info/; http://real-essi.us; http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/). The Real ESSI Simulator (Realistic Modeling and Simulation of Earthquakes, Soils, Structures and their Interaction), (pronunciation of ESSI is similar to easy, as in "as easy as pie") is a software, hardware and documentation system for select level of fidelity (high, low, intermediate), high performance, time domain, nonlinear/inelastic, deterministic or probabilistic, 3D, finite element modeling and simulation of

- statics and dynamics of soil,
- statics and dynamics of rock,
- statics and dynamics of structures,
- statics of soil-structure systems, and
- dynamics of earthquake-soil-structure system interaction.

The Real-ESSI Simulator systems is used for the analysis, design and assessment of static and dynamic behavior of infrastructure objects, including buildings, bridges, dams, nuclear installations, tunnels, etc. The Real-ESSI Simulator develops modeling and simulations that inform and predict rather than (force) fit.

The Real-ESSI Simulator program is available in executable form and through docker container at http://real-essi.info/, http://real-essi.us/. The Real ESSI Simulator system is also available for use on Amazon Web Services Market Place, search for "Real ESSI" or "MS ESSI".

The Real-ESSI is also known, in languages of developers and users, as: Врло просто, Стварно лако, То је лако, Просто к’о пасуљ, Muy fácil, Molto facile, Εύκολο, əsɑ̃n ｗａｑｉ، 本当に簡単, Пряμыатикъ еўклоло, बहुत ही आसान, Très facile, Вистински лесно, Wirklich einfach, Zelo enostavno.
Distribution?

These Lecture Notes can be downloaded and distributed worldwide. If you use these Lecture Notes, please cite them as:


or

## Overview Table of Contents

### 100 Theoretical and Computational Formulations

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>106</td>
<td>Large Deformation Elasto-Plasticity</td>
<td>1996-2004</td>
</tr>
<tr>
<td>107</td>
<td>Solution of Static Equilibrium Equations</td>
<td>1994-2016</td>
</tr>
</tbody>
</table>

### 200 Software and Hardware Platform: Design, Development, Procurement and
## Use

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>210</td>
<td>Software Platform Procurement, Distribution</td>
<td>(2019-2020-2021)</td>
</tr>
</tbody>
</table>

## 300 Verification and Validation

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section</td>
<td>Title</td>
<td>Years</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>308</td>
<td>Verification and Validation for Static and Dynamic Behavior of Special Elements (Contacts/Interfaces/Joints, Gap/Frictional, Isolators)</td>
<td>2010-2011, 2016-2017, 2019-2021</td>
</tr>
<tr>
<td>314</td>
<td>Comparison with Other Programs</td>
<td>2016-2017, 2020-2021</td>
</tr>
</tbody>
</table>

### 400 Education, Training and Modeling, Simulation Examples

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Years</th>
</tr>
</thead>
</table>
Ten Week Course on Dynamic Finite Element Methods for Realistic Modeling and Simulation of Earthquakes, and Soils, and Structures, and their Interaction, Real-ESSI
(1998-2021-)

Online Education and Training
(2019-2020-2021-)

Constitutive, Material Behaviour Examples
(2016-2017-2019-)

Static Examples

Dynamic Examples

Stochastic Examples
(2018-2019-2020-2021-)

Three Day Short Course Examples
(2017-2021-)

Online Short Course Examples
(2019-2020-2021-)

Application to Practical Engineering Problems

Static Soil-Pile and Soil-Pile Group Interaction in Single Phase Soils
(1999-2002-)

Earthquake Soil Structure Interaction, General Aspects

Earthquake-Soil-Structure Interaction, Bridge Structures
(2003-2007-2011-) 

Earthquake-Soil-Structure Interaction, Nuclear Power Plants

Liquefaction and Cyclic Mobility
(2002-2006-2009-2021-)

Slope Stability in 2D and 3D
(1999-2010-)

<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
<th>Dates</th>
</tr>
</thead>
<tbody>
<tr>
<td>507</td>
<td>Concrete Structures</td>
<td>(1989-2017-2018-2020-2021-)</td>
</tr>
<tr>
<td>508</td>
<td>ESSI for Concrete Dams</td>
<td>(2019-2020-2021-)</td>
</tr>
<tr>
<td>511</td>
<td>ASCE-4, Chapter on Nonlinear ESSI analysis</td>
<td>(2016-2020-2021-)</td>
</tr>
<tr>
<td>600</td>
<td>References</td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>701</td>
<td>Useful Formulae</td>
<td>(1985-1989-1993-2021-)</td>
</tr>
<tr>
<td>703</td>
<td>Closed Form Gradients to the Plastic Potential Function</td>
<td>(1993-1994-)</td>
</tr>
<tr>
<td>704</td>
<td>Hyperelasticity, Detailed Derivations</td>
<td>(1995-1996-)</td>
</tr>
<tr>
<td>708</td>
<td>Brief History of the Real-ESSI Simulator Development</td>
<td>(1986-)</td>
</tr>
<tr>
<td>Page</td>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td>709</td>
<td>Work Organization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1989-)</td>
<td></td>
</tr>
<tr>
<td>710</td>
<td>Collected Bibliography</td>
<td></td>
</tr>
</tbody>
</table>
Contents

100 Theoretical and Computational Formulations 86

101 Introduction


101.1 Chapter Summary and Highlights 88

101.2 Specialization to Computational Mechanics 88

101.2.1 Mechanics 88

101.2.2 Continuum Mechanics 88

101.2.3 Statics and Dynamics 88

101.2.4 Discretization Methods 89

101.2.5 The Solution Morass 89

101.2.6 Smooth Nonlinearities 90

101.2.7 Rough Nonlinearities 90

101.3 Tour of Computational Mechanics 90

101.3.1 Equilibrium Path 90

101.3.2 Special Equilibrium Points 90

101.3.2.1 Critical Points 90

101.3.2.2 Turning Points 90

101.3.2.3 Failure Points 90

101.3.3 Generalized Response 90

101.3.4 Sources of Nonlinearities 90

101.3.5 Simulation Process: Loading Stages, Increments and Iterations 90

102 Finite Elements Formulation


102.1 Chapter Summary and Highlights 93

102.2 Formulation of the Continuum Mechanics Incremental Equations of Motion 93
102.3 Finite Element Discretization ........................................ 97
102.3.1 Static Analysis: Internal and External Loads. ............... 106
102.4 Isoparametric Solid Finite Elements ............................... 108
102.4.1 8 Node Brick .................................................. 108
102.4.2 Collapsed 8 Node Brick ........................................ 109
102.4.3 20 Node Brick .................................................. 110
102.4.4 27 Node Brick .................................................. 112
102.4.5 Isoparametric 8 – 20 Node Finite Element ................. 113
102.4.6 Isoparametric 8 - 27 Node Finite Element ................. 116
102.4.7 Surface Loads for Solid Bricks ................................. 118
102.5 Numerical Integration for Solid Brick Elements ............... 120
102.6 Two Node, 3D Truss Finite Element .............................. 120
102.7 3D Beam-Column Finite Element, 12 Degrees of Freedom .... 120
102.8 3D Beam-Column Finite Element, 9 Degrees of Freedom. .. 123
102.9 Shear Beam Finite Element ....................................... 129
102.10 Quadrilateral Shell Finite Element with 6DOFs per Node .... 129
102.11 Seismic Isolator and Dissipator Finite Elements .............. 130
102.11.1 Base Isolation Systems ...................................... 130
102.11.2 Base Dissipator Systems .................................... 131
102.11.3 Two Node, 3D, Rubber Isolator Finite Element .......... 131
102.11.4 Two Node, 3D, Frictional Pendulum Finite Element ..... 131
102.12 Fully Coupled, Porous Solid – Pore Fluid Finite Elements .. 131
102.12.1 u-p-U Formulation ............................................. 131
102.12.1.1 Background ................................................ 131
102.12.1.2 Governing Equations of Porous Media .................. 132
102.12.1.3 Modified Governing Equations. ......................... 134
102.12.1.4 Numerical Solution of the u-p-U Governing Equations . 137
102.12.1.5 Matrix form of the governing equations ............... 142
102.12.1.6 Choice of shape functions ................................ 144
102.12.1.7 8 Node $u – p – U$ Brick ................................ 146
102.12.1.8 20 Node $u – p – U$ Brick ................................ 146
102.12.1.9 27 Node $u – p – U$ Brick ................................ 146
102.12.2 $u-p-U$ Formulation for Partially Saturated, Unsaturated Material 146
102.12.3 $u-p$ Formulation 146
102.12.3.1 Governing Equations of Porous Media 146
102.12.3.2 Numerical Solutions of the Governing Equations 147
102.12.3.3 8 Node $u-p$ Brick 150
102.13 Material and Geometric Non–Linear Finite Element Formulation 150
102.13.1 Introduction 150
102.13.2 Equilibrium Equations 151
102.13.3 Formulation of Non–Linear Finite Element Equations 152
102.13.4 Computational Domain in Incremental Analysis 154
102.13.4.1 Total Lagrangian Format 157
102.13.5 Finite Element Formulations 158
102.13.5.1 Strong Form 158
102.13.5.2 Weak Form 159
102.13.5.3 Linearized Form 159
102.13.5.4 Finite Element Form 160


103.1 Chapter Summary and Highlights 162
103.2 Friction 162
103.2.1 Early Works 162
103.3 Particle Contact Mechanics 162
103.3.1 Particle Contact Mechanics, Axial Behavior 162
103.3.2 Particle Contact Mechanics, Shear Behavior 163
103.4 Dilatancy 163


104.1 Chapter Summary and Highlights 165
104.2 Elasto–plasticity 165
104.2.1 Constitutive Relations for Infinitesimal Plasticity 165
104.2.2 On Integration Algorithms 167
104.2.3 Midpoint Rule Algorithm 168
104.2.3.1 Accuracy Analysis 169
104.2.3.2 Numerical Stability Analysis 175
104.2.4 Crossing the Yield Surface ................................. 185
104.2.5 Singularities in the Yield Surface. .......................... 187
104.2.5.1 Corner Problem ........................................ 187
104.2.5.2 Apex Problem ........................................... 189
104.2.5.3 Influence Regions in Meridian Plane ................. 190
104.3 A Forward Euler (Explicit) Algorithm .................. 192
104.3.1 Continuum Tangent Stiffness Tensor .................. 193
104.4 A Backward Euler (Implicit) Algorithm .................. 193
104.4.1 Single Vector Return Algorithm .......................... 194
104.4.2 Backward Euler Algorithms: Starting Points ....... 197
104.4.2.1 Single Vector Return Algorithm Starting Point .... 197
104.4.3 Consistent Tangent Stiffness Tensor ................... 199
104.4.3.1 Single Vector Return Algorithm ...................... 200
104.4.4 Gradients to the Potential Function ................. 202
104.4.4.1 Analytical Gradients .................................. 202
104.4.4.2 Finite Difference Gradients ......................... 205
104.5 Line Search Technique for Constitutive Elastic-Plastic Integration ....... 207
104.6 Elastic and Elastic–Plastic Material Models for Solids ........ 207
104.6.1 Elasticity .................................................. 207
104.6.1.1 Elastic Model ......................................... 210
104.6.1.2 Non–linear Elastic Model #1 ......................... 210
104.6.1.3 Non–linear Elastic Model #2 ......................... 210
104.6.1.4 Lade's Non–linear Elastic Model .................. 210
104.6.1.5 Cross Anisotropic Linear Elastic Model .......... 211
104.6.2 Yield Functions ......................................... 211
104.6.3 Plastic Flow Directions ................................. 211
104.6.4 Hardening–Softening Evolution Laws ............... 212
104.6.5 Tresca Model ........................................... 212
104.6.6 von Mises Model ....................................... 213
104.6.6.1 Yield and Plastic Potential Functions: von Mises Model (form I) .... 216
104.6.6.2 Yield and Plastic Potential Functions: von Mises Model (form II) .... 217
104.6.6.3 Hardening and Softening Functions: von Mises Model .......... 218
104.6.7 Drucker-Prager Model ........................................... 220
  104.6.7.1 Yield and Plastic Potential Functions: Drucker-Prager Model (form I) ........................................... 222
  104.6.7.2 Yield and Plastic Potential Functions: Drucker-Prager Model (form II) ........................................... 223
  104.6.7.3 Hardening and Softening Functions: Drucker-Prager Model ........................................... 224
  104.6.7.4 Federico's Description of a Drucker–Prager Kinematic Hardening Model ........................................... 226
  104.6.7.5 Han's Description of Drucker–Prager Model with Armstrong–Frederick Kinematic Hardening ........................................... 228
104.6.8 Hyperbolic Drucker Prager Model ........................................... 231
  104.6.8.1 Original Yield Function and Hyperbolic Function ........................................... 231
  104.6.8.2 Hyperbolic Drucker Prager Model ........................................... 231
  104.6.8.3 Modified Hyperbolic Drucker Prager ........................................... 232
  104.6.8.4 The Non-Associative Plastic Potential Function ........................................... 233
  104.6.8.5 Han's Description of Hyperbolic Drucker–Prager Model with Armstrong–Frederick Kinematic Hardening ........................................... 234
104.6.9 Rounded Mohr-Coulomb Model ........................................... 237
104.6.10 Modified Cam-Clay Model ........................................... 238
  104.6.10.1 Critical State ........................................... 239
  104.6.10.2 Elasticity ........................................... 239
  104.6.10.3 Yield Function ........................................... 240
  104.6.10.4 Plastic Flow ........................................... 240
  104.6.10.5 Evolution Law ........................................... 241
  104.6.10.6 Yield and Plastic Potential Functions: Cam-Clay Model. ........................................... 241
  104.6.11 SaniSand2004 (aka Dafalias-Manzari) Model ........................................... 242
    104.6.11.1 Critical State ........................................... 242
    104.6.11.2 Elasticity ........................................... 243
    104.6.11.3 Yield Function ........................................... 243
    104.6.11.4 Plastic Flow ........................................... 245
    104.6.11.5 Evolution Laws ........................................... 245
    104.6.11.6 Analytical Derivatives for the Implicit Algorithm ........................................... 246
  104.6.12 SaniSand2008 (aka SANISAND) Model ........................................... 249
  104.6.13 SANICLAY Model ........................................... 249
  104.6.14 Pisanò Elastic-Plastic Model with Vanishing Elastic Region (for $G/G_{\text{max}}$ Modeling) ........................................... 249
     104.6.14.1 Frictional and viscous dissipative mechanisms ........................................... 251
104.8.1.4 High Rate Elastic-Plastic Loading .............................................. 324
104.8.1.5 Coupling with Pore Fluid Pressure and Temperature .................. 328
104.8.2 Uncertainty and Variability of Rock Behavior .............................. 333
104.8.3 Effects of Shock Loading on Intact Rock Behavior ....................... 334
104.8.3.1 Shock Waves of First and Second Kind ...................................... 335
104.8.3.2 Hugonions ................................................................. 337
104.8.4 Material Modeling of Rock ......................................................... 338
104.8.4.1 Lawrence Livermore National Laboratory Models ....................... 338
104.8.4.2 Hoek and Brown Model ...................................................... 338
104.8.4.3 Other Models ............................................................... 339
104.8.5 Model Calibration / Testing Devices ............................................. 341
104.8.6 Influence of Pore Fluid Pressure and Temperature on Rock Response .. 343
104.9 Inelastic Behavior and Models for Concrete Beams, Walls and Shells .... 343
104.9.1 Uniaxial Material Model for Steel ................................................ 343
104.9.2 3D Plastic Damage Concrete Material Model, Faria-Oliver-Cervera .... 345
104.10 Calibration of Elastic-Plastic Material Models ................................. 348
104.10.1 Calibration of Elastic-Plastic Material Models, Soil ......................... 348
104.10.2 Calibration of Elastic-Plastic Material Models, Rock ....................... 348
104.10.3 Calibration of Elastic-Plastic Material Models, Contact/Joint/Interface .. 348
104.10.4 Calibration of Elastic-Plastic Material Models, Concrete ................ 348
104.10.5 Calibration of Elastic-Plastic Material Models, Steel ....................... 348
104.11 Energy Dissipation Calculations for Solids ..................................... 348
104.11.1 Introduction ........................................................................ 348
104.11.2 Theoretical and Computational Formulations ................................. 351
104.11.2.1 Thermo-Mechanical Theory .................................................. 351
104.11.2.2 Plastic Free Energy ........................................................... 352
104.11.2.3 Plastic Dissipation ............................................................... 355
104.11.2.4 Energy Computation in Finite Elements .................................... 357
104.11.3 Numerical Studies ................................................................. 359
104.11.3.1 Elastic Material ............................................................... 359
104.11.3.2 von Mises Plasticity ............................................................ 361
104.11.3.3 Drucker–Prager Plasticity ..................................................... 367
104.11.4 Conclusions ........................................................................ 371
104.12 Energy Dissipation Calculations for Structures

104.12.1 Introduction

104.12.2 Theoretical and Computational Formulations

104.12.2.1 Thermomechanical Framework

104.12.2.2 Plastic Free Energy

104.12.2.3 Energy Dissipation in Beam-Column Element

104.12.3 Numerical Studies

104.12.3.1 Steel Column

104.12.3.2 Plain Concrete Column

104.12.3.3 Reinforced Concrete Column

104.12.3.4 Steel Frame

104.12.4 Conclusions

105 Probabilistic Elasto-Plasticity and Stochastic Elastic-Plastic Finite Element Method

105.1 Chapter Summary and Highlights

105.2 Probabilistic Elasto-Plasticity

105.2.1 Probabilistic Elasto-Plasticity: Introduction

105.2.2 Probabilistic Elasto-Plasticity: General Formulation

105.2.3 Probabilistic Elasto-Plasticity: Elastic–Plastic Probabilistic 1-D Constitutive Incremental Equation

105.2.4 Probabilistic Elasto-Plasticity: Initial and Boundary Conditions for the Probabilistic Elastic–Plastic PDE

105.2.5 Probabilistic Elasto-Plasticity: Fokker–Planck–Kolmogorov Equation for Probabilistic Elasticity and Elasto–Plasticity in 1-D

105.2.6 Probabilistic Elasto-Plasticity: Example Problem Statements

105.2.7 Probabilistic Elasto-Plasticity: Determination of Coefficients for Fokker–Planck–Kolmogorov Equation

105.2.8 Probabilistic Elasto-Plasticity: Results and Verifications of Example Problems

105.2.9 Problem I

105.2.10 Problem II

105.2.11 Problem III
105.3 Probabilistic Yielding and Cyclic Loading

105.3.1 Fokker–Planck–Kolmogorov Approach to Probabilistic Elasto–Plasticity

105.3.2 Elastic–Perfectly Plastic Material

105.3.2.1 Probability Density Function

105.3.2.2 Case of Increasing Strain Loops

105.3.2.3 Case of Constant Strain Loops

105.3.2.4 Monotonic Loading

105.3.2.5 Hardening Material

105.3.2.6 Isotropic Hardening

105.3.2.7 Kinematic Hardening

105.4 Stochastic Elastic-Plastic Finite Element Method

106 Large Deformation Elasto-Plasticity

106.1 Chapter Summary and Highlights

106.2 Continuum Mechanics Preliminaries: Kinematics

106.2.1 Deformation

106.2.2 Deformation Gradient

106.2.3 Strain Tensors, Deformation Tensors and Stretch

106.2.4 Rate of Deformation Tensor

106.3 Constitutive Relations: Hyperelasticity

106.3.1 Introduction

106.3.2 Isotropic Hyperelasticity

106.3.3 Volumetric–Isochoric Decomposition of Deformation

106.3.4 Simo–Serrin’s Formulation

106.3.5 Stress Measures

106.3.6 Tangent Stiffness Operator

106.3.7 Isotropic Hyperelastic Models

106.3.7.1 Ogden Model

106.3.7.2 Neo–Hookean Model

106.3.7.3 Mooney–Rivlin Model

106.3.7.4 Logarithmic Model

106.3.7.5 Simo–Pister Model
106.4 Finite Deformation Hyperelasto–Plasticity ........................................ 474
   106.4.1 Introduction ................................................................. 474
   106.4.2 Kinematics ................................................................. 474
   106.4.3 Constitutive Relations .................................................... 476
   106.4.4 Implicit Integration Algorithm ........................................... 477
   106.4.5 Algorithmic Tangent Stiffness Tensor ................................. 487

107 Solution of Static Equilibrium Equations
   (1994-2016-) ..................................................................... 491
   107.1 Chapter Summary and Highlights ........................................... 492
   107.2 The Residual Force Equations .............................................. 492
   107.3 Constraining the Residual Force Equations ............................ 492
   107.4 Load Control ................................................................. 495
   107.5 Displacement Control ......................................................... 495
   107.6 Generalized, Hyper–Spherical Arc-Length Control ................. 495
      107.6.1 Traversing Equilibrium Path in Positive Sense .................. 498
         107.6.1.1 Positive External Work ........................................... 498
         107.6.1.2 Angle Criterion ....................................................... 499
      107.6.2 Predictor step .............................................................. 500
      107.6.3 Automatic Increments ................................................... 501
      107.6.4 Convergence Criteria ................................................... 501
      107.6.5 The Algorithm Progress .............................................. 505

108 Solution of Dynamic Equations of Motion
   108.1 Chapter Summary and Highlights ........................................... 507
   108.2 The Principle of Virtual Displacements in Dynamics ................ 507
   108.3 Direct Integration Methods for the Equations of Dynamic Equilibrium ................................................. 507
      108.3.1 Newmark Integrator ..................................................... 507
      108.3.2 HHT Integrator ............................................................ 508
   108.4 Synthetic Viscous Damping for Solids and Structures ............. 509
      108.4.1 Synthetic Viscous Damping Approaches .......................... 509
      108.4.2 Caughey Damping 2nd Order, aka Rayleigh Damping .......... 511
      108.4.3 Caughey Damping 3rd Order ........................................ 513
      108.4.4 Caughey Damping 4th Order ........................................ 513
109 Earthquake Soil Structure Interaction, Theoretical Aspects

109.1 Chapter Summary and Highlights
109.2 Seismic Energy Propagation and Dissipation
  109.2.1 Seismic energy input into SSI system
  109.2.2 Seismic Energy Dissipation in SSI System
  109.2.2.1 Energy Dissipation by Plasticity
  109.2.2.2 Energy Dissipation by Viscous Coupling
  109.2.2.3 Numerical Energy Dissipation and Production
  109.2.2.4 Energy Dissipation by Nonlinearities in Soil/Rock
  109.2.2.5 Energy Dissipation by Nonlinearities in Soil/Rock – Foundation Interface Zone
  109.2.2.6 Energy Dissipation by Nonlinearities in Seismic Isolators
  109.2.2.7 Energy Dissipation by Nonlinearities in Structures, Systems and Components
  109.2.2.8 Numerical Energy Dissipation and Production
  109.2.3 Seismic Motions: Empirical Models
  109.2.4 1D/1C Wave Propagation Modeling
  109.2.5 Seismic Motions: 3D/3C Analytic Wave Propagation Modeling
  109.2.6 Seismic Motions: Large Scale Geophysical Models
  109.2.6.1 Regional Seismic Motion Modeling using Serpentine Wave Propagation, SW4
  109.2.7 Site Response
  109.2.8 Seismic Motion Incoherence
  109.2.8.1 Lack of Correlation Modeling and Simulation
  109.2.9 Lack of Volume Change Data for Soil
  109.3 Earthquake Soil Structure Interaction
  109.4 Earthquake Soil Structure Interaction Modeling Details
  109.4.1 Seismic Motions Input into Finite Element Model
  109.4.1.1 The Domain Reduction Method (DRM) Development

110 Parallel Computing in Computational Mechanics

110.1 Chapter Summary and Highlights
110.2 Introduction ........................................ 548
110.2.1 High Performance Computing on DMPs, SMPs, GPGPUs, FPGA .... 548
110.2.1.1 Distributed Memory Parallel (SMP) Computations .................. 548
110.2.1.2 Shared Memory Parallel (SMP) Computations ....................... 548
110.2.1.3 General Purpose Graphical Processiong Units (GPGPUs) .......... 548
110.2.1.4 Fast Programmable Gate Arrays (FPGAs) ......................... 548
110.2.2 Parallel Computing for Elastic-Plastic Solids and Structures .......... 548
110.2.3 Problem Requirements .................................. 548
110.2.3.1 Finite Element Computations in Geomechanics ..................... 548
110.2.3.2 Adaptive Computation ......................................... 550
110.2.3.3 Multi-phase Computation ...................................... 551
110.2.4 Parallel Computing Hardware .................................. 551
110.2.4.1 DMPs and SMPs ................................................ 551
110.2.5 Parallel Computing Software .................................... 552
110.2.5.1 Amdahl’s Law .................................................. 552
110.2.5.2 Static and Dynamic Graph Partitioning ............................ 552
110.2.5.3 Real parallel and embarrassingly parallel ......................... 552
110.2.5.4 Parallel Computing for Elastic-Plastic FEM ....................... 553
110.2.5.5 Plastic Domain Decomposition .................................. 553
110.2.5.6 Template Meta-programs ..................................... 554
110.3 Plastic Domain Decomposition Algorithm ............................. 554
110.3.1 Introduction ................................................. 554
110.3.2 Inelastic Parallel Finite Element .................................... 556
110.3.2.1 Adaptive Computation ......................................... 558
110.3.2.2 Multiphase Computation ...................................... 559
110.3.2.3 Multiconstraint Graph Partitioning ............................. 559
110.3.2.4 Adaptive PDD Algorithm .................................... 561
110.3.3 Adaptive Multilevel Graph Partitioning Algorithm ................... 561
110.3.3.1 Unified Repartitioning Algorithm ............................. 566
110.3.3.2 Study of ITR in ParMETIS .................................. 567
110.4 Performance Studies on PDD Algorithm ............................... 568
110.4.1 Introduction ................................................. 568
110.4.2 Parallel Computers ............................................ 568
110.4.3 Soil-Foundation Interaction Model ................................... 569
110.4.4 Numerical Study for ITR. ........................................ 571
110.4.5 Parallel Performance Analysis .................................. 581
110.4.5.1 Soil-Foundation Model with 4,035 DOFs .................... 582
110.4.5.2 Soil-Foundation Model with 4,938 Elements, 17,604 DOFs 587
110.4.5.3 Soil-Foundation Model with 9,297 Elements, 32,091 DOFs 592
110.4.6 Algorithm Fine-Tuning ............................................. 598
110.4.7 Fine Tuning on Load Imbalance Tolerance .................... 599
110.4.8 Globally Adaptive PDD Algorithm ............................. 603
110.4.8.1 Implementations .................................................. 605
110.4.8.2 Performance Results ............................................ 606
110.4.9 Scalability Study on Prototype Model ......................... 611
110.4.9.1 3 Bent SFSI Finite Element Models ......................... 611
110.4.9.2 Scalability Runs .................................................. 613
110.4.10 Conclusions ........................................................ 614

110.5 Application of Project-Based Iterative Methods in SFSI Problems .... 621
110.5.1 Introduction .......................................................... 621
110.5.2 Projection-Based Iterative Methods ......................... 621
110.5.2.1 Conjugate Gradient Algorithm ........................... 622
110.5.2.2 GMRES ......................................................... 623
110.5.2.3 BiCGStab and QMR ........................................... 625
110.5.3 Preconditioning Techniques ..................................... 625
110.5.4 Preconditioners ..................................................... 627
110.5.4.1 Jacobi Preconditioner ...................................... 627
110.5.4.2 Incomplete Cholesky Preconditioner ...................... 627
110.5.4.3 Robust Incomplete Factorization ........................... 628
110.5.5 Numerical Experiments .......................................... 631
110.5.6 Conclusion and Future Work ................................... 639

110.6 Performance Study on Parallel Direct/Iterative Solving in SFSI .... 640
110.6.1 Parallel Sparse Direct Equation Solvers ....................... 643
110.6.1.1 General Techniques – SPOOLES ........................ 643
110.6.1.2 Frontal and Multifrontal Methods – MUMPS. .......... 643
110.6.1.3 Supernodal Algorithm – SuperLU ....................... 646
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>110.6.2</td>
<td>Performance Study on SFSI Systems</td>
<td>648</td>
</tr>
<tr>
<td>110.6.2.1</td>
<td>Equation System</td>
<td>648</td>
</tr>
<tr>
<td>110.6.2.2</td>
<td>Performance Results</td>
<td>650</td>
</tr>
<tr>
<td>110.6.3</td>
<td>Conclusion</td>
<td>650</td>
</tr>
<tr>
<td>111</td>
<td>Solid, Structure – Fluid Interaction</td>
<td>652</td>
</tr>
<tr>
<td>111.1</td>
<td>Chapter Summary and Highlights</td>
<td>653</td>
</tr>
<tr>
<td>111.2</td>
<td>Introduction</td>
<td>653</td>
</tr>
<tr>
<td>111.3</td>
<td>Theoretical Formulation</td>
<td>655</td>
</tr>
<tr>
<td>111.3.1</td>
<td>Solid Fluid Interaction</td>
<td>655</td>
</tr>
<tr>
<td>111.3.2</td>
<td>Finite Volume Discretization</td>
<td>656</td>
</tr>
<tr>
<td>111.3.3</td>
<td>Volume of Fluid Method</td>
<td>659</td>
</tr>
<tr>
<td>111.3.4</td>
<td>Pressure-velocity coupling: PISO algorithm</td>
<td>660</td>
</tr>
<tr>
<td>111.3.5</td>
<td>Explicit transient algorithm</td>
<td>663</td>
</tr>
<tr>
<td>111.4</td>
<td>Implementation Details</td>
<td>665</td>
</tr>
<tr>
<td>111.4.1</td>
<td>Installation of OpenFoam</td>
<td>665</td>
</tr>
<tr>
<td>111.4.2</td>
<td>Integrated Preprocessor-gmFoam</td>
<td>665</td>
</tr>
<tr>
<td>111.4.3</td>
<td>Interface Domain-SSFI</td>
<td>667</td>
</tr>
<tr>
<td>111.4.4</td>
<td>Geometric Mapping</td>
<td>668</td>
</tr>
<tr>
<td>111.4.5</td>
<td>SFI Interpolation</td>
<td>669</td>
</tr>
<tr>
<td>111.4.6</td>
<td>Mass Conservation</td>
<td>672</td>
</tr>
<tr>
<td>200</td>
<td>Software and Hardware Platform: Design, Development, Procurement and Use</td>
<td>674</td>
</tr>
<tr>
<td>201</td>
<td>The Real ESSI Simulator System</td>
<td>675</td>
</tr>
<tr>
<td>201.1</td>
<td>Chapter Summary and Highlights</td>
<td>676</td>
</tr>
<tr>
<td>201.2</td>
<td>Introduction to the Real-ESSI Simulator System</td>
<td>676</td>
</tr>
<tr>
<td>201.2.1</td>
<td>Real-ESSI Program</td>
<td>676</td>
</tr>
<tr>
<td>201.2.2</td>
<td>Real-ESSI Pre-Processing tools</td>
<td>677</td>
</tr>
<tr>
<td>201.2.3</td>
<td>Real-ESSI Post-Processing tools</td>
<td>677</td>
</tr>
<tr>
<td>201.2.4</td>
<td>Real-ESSI Computer</td>
<td>677</td>
</tr>
<tr>
<td>201.2.5</td>
<td>Real-ESSI Notes</td>
<td>677</td>
</tr>
</tbody>
</table>
202 Object Oriented Software Platform Design


202.1 Chapter Summary and Highlights

202.2 Object-Oriented Design Basics

202.3 Object-Oriented Design of the Plastic Domain Decomposition (PDD).

202.3.1 Introduction

202.3.2 Object-Oriented Parallel Finite Element Algorithm

202.3.2.1 Modeling Classes

202.3.2.2 Finite Element Model Class

202.3.2.3 Analysis

202.3.2.4 Object-Oriented Domain Decomposition

202.3.2.5 Parallel Object-Oriented Finite Element Design

202.3.3 Dual-Phase Adaptive Load Balancing

202.3.3.1 Elemental Level Load Balancing

202.3.3.2 Equation Solving Load Balancing

202.3.4 Object-Oriented Design of PDD

202.3.4.1 MPI_Channel

202.3.4.2 MPI_ChannelAddress

202.3.4.3 FEM_ObjectBroker

202.3.4.4 Domain

202.3.4.5 PartitionedDomain

202.3.4.6 Node & DOF_Group

202.3.4.7 DomainPartitioner

202.3.4.8 Shadow/ActorSubdomain

202.3.4.9 Send/RecvSelf

202.3.5 Graph Partitioning

202.3.5.1 Construction of Element Graph

202.3.5.2 Interface to ParMETIS/METIS

202.3.6 Data Redistribution
203 Library Centric Software Platform Design


203.1 Chapter Summary and Highlights ....................................................... 721

203.1.1 Finite Elements .............................................................................. 721

203.1.1.1 Single Phase Solid Elements ..................................................... 721

203.1.1.2 Fully Coupled, Two-Phase (Porous Solid – Pore Fluid) Solid Elements . 721

203.1.1.3 Structural Elements ................................................................ 721

203.1.1.4 Special Elements .................................................................... 721

203.1.2 Constitutive Integration and Material Models .................................. 722

203.1.2.1 Explicit Integration .................................................................. 722

203.1.2.2 Implicit Integration .................................................................. 722

203.1.2.3 Material Models ...................................................................... 722

203.1.3 Modified OpenSees Services Library .............................................. 722

204 Application Programming Interface


204.1 Chapter Summary and Highlights ....................................................... 724

204.2 Introduction ....................................................................................... 724

204.3 Application Programming Interface for Domain Specific Language (DSL) .. 724

204.3.1 Modeling ....................................................................................... 724

204.3.1.1 Modeling: Material Models ..................................................... 725

204.3.1.2 Modeling: Nodes .................................................................... 737

204.3.1.3 Modeling: Finite Elements ..................................................... 738

204.3.1.4 Modeling: Damping ................................................................. 752

204.3.1.5 Modeling: Constraints, Supports, Tied Nodes Connections, etc. .. 753

204.3.1.6 Modeling: Static Loads ............................................................ 755

204.3.1.7 Modeling: Dynamic Loads ....................................................... 759

204.3.1.8 Modeling: Prescribed Displacements ...................................... 761

204.3.1.9 Solid-Fluid Interface ............................................................... 761

204.3.1.10 Outputs to mySQL database .................................................. 762

204.3.2 Simulation ...................................................................................... 762

204.3.2.1 Simulation: Solvers .................................................................. 762

204.3.2.2 Simulation: Static Solution Advancement .................................. 763

204.3.2.3 Simulation: Dynamic Solution Advancement ........................... 763

204.3.2.4 Simulation: Solution Algorithms ............................................ 764
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>204.3.2.5 Simulation: Convergence Criteria</td>
<td>764</td>
</tr>
<tr>
<td>204.3.2.6 Simulating Response</td>
<td>765</td>
</tr>
<tr>
<td>204.4 Application Programming Interface for Constitutive Simulations</td>
<td>766</td>
</tr>
<tr>
<td>204.5 Application Programming Interface for Finite Elements</td>
<td>766</td>
</tr>
<tr>
<td>204.6 Adding New Finite Element into Real-ESSI Simulator</td>
<td>767</td>
</tr>
<tr>
<td>204.6.1 Introduction</td>
<td>767</td>
</tr>
<tr>
<td>204.6.2 Getting Started:: Creating New Element Directory</td>
<td>767</td>
</tr>
<tr>
<td>204.6.3 Element Header File</td>
<td>769</td>
</tr>
<tr>
<td>204.6.4 Element Source File</td>
<td>772</td>
</tr>
<tr>
<td>204.6.5 Element Class Tag Description</td>
<td>781</td>
</tr>
<tr>
<td>204.6.6 Integrating New Finite Element into Parser</td>
<td>784</td>
</tr>
<tr>
<td>204.6.6.1 feiparser.yy</td>
<td>785</td>
</tr>
<tr>
<td>204.6.6.2 feiparser.l</td>
<td>785</td>
</tr>
<tr>
<td>204.6.6.3 Argument Stack, Signature and Units</td>
<td>785</td>
</tr>
<tr>
<td>204.6.6.4 FeiDslCaller</td>
<td>787</td>
</tr>
<tr>
<td>204.6.6.5 New DSL Header File</td>
<td>787</td>
</tr>
<tr>
<td>204.6.7 Compiling Real-ESSI</td>
<td>788</td>
</tr>
<tr>
<td>204.6.8 Verification of Implementation</td>
<td>788</td>
</tr>
<tr>
<td>205 Input, Domain Specific Language</td>
<td>789</td>
</tr>
<tr>
<td>205.1 Chapter Summary and Highlights</td>
<td>790</td>
</tr>
<tr>
<td>205.2 Introduction</td>
<td>790</td>
</tr>
<tr>
<td>205.3 Domain Specific Language (DSL), English Language Binding</td>
<td>791</td>
</tr>
<tr>
<td>205.3.1 Running Real-ESSI</td>
<td>791</td>
</tr>
<tr>
<td>205.3.2 Finishing Real-ESSI Program Run</td>
<td>793</td>
</tr>
<tr>
<td>205.3.3 Real-ESSI Variables, Basic Units and Flow Control</td>
<td>794</td>
</tr>
<tr>
<td>205.3.4 Modeling</td>
<td>798</td>
</tr>
<tr>
<td>205.3.4.1 Modeling, Material Model: Adding a Material Model to the Finite Element Model</td>
<td>800</td>
</tr>
<tr>
<td>205.3.4.2 Modeling, Material Model: Linear Elastic Isotropic Material Model</td>
<td>801</td>
</tr>
<tr>
<td>205.3.4.3 Modeling, Material Model: Cross Anisotropic Linear Elastic Material Model</td>
<td>802</td>
</tr>
<tr>
<td>205.3.4.4 Modeling, Material Model: von Mises Associated Material Model with Linear Isotropic and/or Kinematic Hardening</td>
<td>803</td>
</tr>
<tr>
<td>205.3.4.5</td>
<td>Modeling, Material Model: von Mises Associated Material Model with Isotropic Hardening and/or Armstrong-Frederic Nonlinear Kinematic Hardening</td>
</tr>
<tr>
<td>205.3.4.6</td>
<td>Modeling, Material Model: Drucker-Prager Associated Material Model with Linear Isotropic and/or Kinematic Hardening</td>
</tr>
<tr>
<td>205.3.4.7</td>
<td>Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederic Nonlinear Kinematic Hardening</td>
</tr>
<tr>
<td>205.3.4.8</td>
<td>Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederic Nonlinear Kinematic Hardening and Nonlinear Duncan-Chang Elasticity</td>
</tr>
<tr>
<td>205.3.4.9</td>
<td>Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Kinematic Hardening</td>
</tr>
<tr>
<td>205.3.4.10</td>
<td>Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Armstrong-Frederic Nonlinear Kinematic Hardening</td>
</tr>
<tr>
<td>205.3.4.11</td>
<td>Modeling, Material Model: Hyperbolic Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Armstrong-Frederic Nonlinear Kinematic Hardening</td>
</tr>
<tr>
<td>205.3.4.12</td>
<td>Modeling, Material Model: Rounded Mohr-Coulomb Associated Linear Isotropic Hardening Material Model</td>
</tr>
<tr>
<td>205.3.4.13</td>
<td>Modeling, Material Model: Cam Clay Material Model</td>
</tr>
<tr>
<td>205.3.4.14</td>
<td>Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model</td>
</tr>
<tr>
<td>205.3.4.15</td>
<td>Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model that Matches $G/G_{\text{max}}$ Curves</td>
</tr>
<tr>
<td>205.3.4.16</td>
<td>Modeling, Material Model: Drucker-Prager Nonassociated Multi-Yield Surface Material Model</td>
</tr>
<tr>
<td>205.3.4.17</td>
<td>Modeling, Material Model: Drucker-Prager Nonassociated Material Model that Matches $G/G_{\text{max}}$ Curves</td>
</tr>
<tr>
<td>205.3.4.18</td>
<td>Modeling, Material Model: Rounder Mohr-Coulomb Nonassociated Multi-Yield Surface Material Model</td>
</tr>
<tr>
<td>205.3.4.19</td>
<td>Modeling, Material Model: Tsinghua Liquefaction Material Model</td>
</tr>
<tr>
<td>205.3.4.20</td>
<td>Modeling, Material Model: SANISand Material Model, version 2004</td>
</tr>
<tr>
<td>205.3.4.21</td>
<td>Modeling, Material Model: SANISand Material Model, version 2008</td>
</tr>
<tr>
<td>205.3.4.22</td>
<td>Modeling, Material Model: Cosserat Linear Elastic Material Model</td>
</tr>
<tr>
<td>205.3.4.23</td>
<td>Modeling, Material Model: von Mises Cosserat Material Model</td>
</tr>
<tr>
<td>205.3.4.24</td>
<td>Modeling, Material Model: Uniaxial Linear Elastic, Fiber Material Model</td>
</tr>
<tr>
<td>205.3.4.25</td>
<td>Modeling, Material Model: Stochastic Uniaxial Linear Elastic Model</td>
</tr>
<tr>
<td>205.3.4.26</td>
<td>Modeling, Material Model: Stochastic Uniaxial Nonlinear Armstrong Frederick Model</td>
</tr>
<tr>
<td>205.3.4.27</td>
<td>Modeling, Material Model: Uniaxial Nonlinear Concrete, Fiber Material Model, version 02</td>
</tr>
<tr>
<td>205.3.4.28</td>
<td>Modeling, Material Model: Faria-Oliver-Cervera Concrete Material</td>
</tr>
<tr>
<td>205.3.4.29</td>
<td>Modeling, Material Model: Plane Stress Layered Material</td>
</tr>
<tr>
<td>205.3.4.30</td>
<td>Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 01</td>
</tr>
<tr>
<td>205.3.4.31</td>
<td>Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 02</td>
</tr>
<tr>
<td>205.3.4.32</td>
<td>Modeling, Material Model: Plane Stress Plastic Damage Concrete Material</td>
</tr>
<tr>
<td>205.3.4.33</td>
<td>Modeling, Material Model: Plane Stress Rebar Material</td>
</tr>
<tr>
<td>205.3.4.34</td>
<td>Modeling, Nodes: Adding Nodes</td>
</tr>
<tr>
<td>205.3.4.35</td>
<td>Modeling, Nodes: Adding Stochastic Nodes</td>
</tr>
<tr>
<td>205.3.4.36</td>
<td>Modeling, Nodes: Define Nodal Physical Group</td>
</tr>
<tr>
<td>205.3.4.37</td>
<td>Modeling, Nodes: Adding Nodes to Nodal Physical Group</td>
</tr>
<tr>
<td>205.3.4.38</td>
<td>Modeling, Nodes: Removing Nodal Physical Group</td>
</tr>
<tr>
<td>205.3.4.39</td>
<td>Modeling, Nodes: Print Nodal Physical Group</td>
</tr>
<tr>
<td>205.3.4.40</td>
<td>Modeling, Nodes: Removing Nodes</td>
</tr>
<tr>
<td>205.3.4.41</td>
<td>Modeling, Nodes: Adding Nodal Mass, for 3DOFs and/or 6DOFs</td>
</tr>
<tr>
<td>205.3.4.42</td>
<td>Modeling, Finite Element: Adding Finite Elements</td>
</tr>
<tr>
<td>205.3.4.43</td>
<td>Modeling, Finite Element: Define Finite Element Physical Group</td>
</tr>
<tr>
<td>205.3.4.44</td>
<td>Modeling, Finite Element: Adding Elements to Physical Element Group</td>
</tr>
<tr>
<td>205.3.4.45</td>
<td>Modeling, Finite Element: Remove Physical Finite Element Group</td>
</tr>
<tr>
<td>205.3.4.46</td>
<td>Modeling, Finite Element: Print Physical Finite Element Group</td>
</tr>
<tr>
<td>205.3.4.47</td>
<td>Modeling, Finite Element: Remove Finite Element</td>
</tr>
<tr>
<td>205.3.4.48</td>
<td>Modeling, Finite Element: Truss Element</td>
</tr>
<tr>
<td>205.3.4.49</td>
<td>Modeling, Finite Element: Kelvin-Voigt Truss Element</td>
</tr>
</tbody>
</table>
205.3.4.50 Modeling, Finite Element: Shear Beam Element ........................................ 860
205.3.4.51 Modeling, Finite Element: Stochastic Shear Beam Element .................. 861
205.3.4.52 Modeling, Finite Element: Elastic Beam–Column Element ...................... 862
205.3.4.53 Modeling, Finite Element: Large Displacement Elastic Beam–Column Element, with Corotational Transformation .................................................. 865
205.3.4.54 Modeling, Finite Element: Timoshenko Elastic Beam–Column Element .... 866
205.3.4.55 Modeling, Finite Element: Timoshenko Elastic Beam–Column Element with Directional Shear Correction Coefficients ................................................. 868
205.3.4.56 Modeling, Finite Element: Adding 1D Fiber to a Beam Cross Section ....... 870
205.3.4.57 Modeling, Finite Element: Adding Fiber Section to the Finite Element Model ................................................................. 871
205.3.4.58 Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element ................................................................. 872
205.3.4.59 Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element with Corotational Coordinate Transformation .......................................... 873
205.3.4.60 Modeling, Finite Element: 3DOF+6DOF=9DOF Beam-Column Element .... 874
205.3.4.61 Modeling, Finite Element: 4 Node ANDES Shell with Drilling DOFs ........ 876
205.3.4.62 Modeling, Finite Element: 3 Node ANDES Shell with Drilling DOFs ........ 877
205.3.4.63 Modeling, Finite Element: 4 Node Shell NLDKGQ, or 4 Node Shell Xin-Zheng-Lu ................................................................. 878
205.3.4.64 Modeling, Finite Element: Inelastic Layered Shell Section ...................... 879
205.3.4.65 Modeling, Finite Element: ElasticMembranePlaneStress Element (to be removed!) ................................................................. 880
205.3.4.66 Modeling, Finite Element: InelasticMembranePlaneStress Element (to be removed!) ................................................................. 881
205.3.4.67 Modeling, Finite Element: SuperElementLinearElasticImport ................. 882
205.3.4.68 Modeling, Finite Element: 8 Node Brick Element ...................................... 884
205.3.4.69 Modeling, Finite Element: 20 Node Brick Element .................................... 886
205.3.4.70 Modeling, Finite Element: 27 Node Brick Element .................................... 888
205.3.4.71 Modeling, Finite Element: Variable 8-27 Node Brick Element ................. 890
205.3.4.72 Modeling, Finite Element: 8 Node Brick u-p Element .............................. 892
205.3.4.73 Modeling, Finite Element: 20 Node Brick u-p Element .............................. 895
205.3.4.74 Modeling, Finite Element: 27 Node Brick u-p Element .............................. 898
205.3.4.75 Modeling, Finite Element: 8 Node Brick u-p-U Element ............................ 901
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>205.3.4.76</td>
<td>Modeling, Finite Element: 20 Node Brick u-p-U Element</td>
<td>904</td>
</tr>
<tr>
<td>205.3.4.77</td>
<td>Modeling, Finite Element: 27 Node Brick u-p-U Element</td>
<td>907</td>
</tr>
<tr>
<td>205.3.4.78</td>
<td>Modeling, Finite Element: 8 Node Cosserat Brick Element</td>
<td>910</td>
</tr>
<tr>
<td>205.3.4.79</td>
<td>Modeling, Finite Element: Bonded Contact/Interface/Joint Element</td>
<td>911</td>
</tr>
<tr>
<td>205.3.4.80</td>
<td>Modeling, Finite Element: Force Based Dry Hard Contact/Interface/Joint Element</td>
<td>912</td>
</tr>
<tr>
<td>205.3.4.81</td>
<td>Modeling, Finite Element: Force Based Dry Soft Contact/Interface/Joint Element</td>
<td>913</td>
</tr>
<tr>
<td>205.3.4.82</td>
<td>Modeling, Finite Element: Force Based Coupled Hard Contact/Interface/Joint Element</td>
<td>915</td>
</tr>
<tr>
<td>205.3.4.83</td>
<td>Modeling, Finite Element: Force Based Coupled Soft Contact/Interface/Joint Element</td>
<td>917</td>
</tr>
<tr>
<td>205.3.4.84</td>
<td>Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>919</td>
</tr>
<tr>
<td>205.3.4.85</td>
<td>Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior</td>
<td>921</td>
</tr>
<tr>
<td>205.3.4.86</td>
<td>Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior</td>
<td>923</td>
</tr>
<tr>
<td>205.3.4.87</td>
<td>Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>926</td>
</tr>
<tr>
<td>205.3.4.88</td>
<td>Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior</td>
<td>928</td>
</tr>
<tr>
<td>205.3.4.89</td>
<td>Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior</td>
<td>930</td>
</tr>
<tr>
<td>205.3.4.90</td>
<td>Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>933</td>
</tr>
<tr>
<td>205.3.4.91</td>
<td>Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior</td>
<td>935</td>
</tr>
<tr>
<td>205.3.4.92</td>
<td>Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior</td>
<td>937</td>
</tr>
<tr>
<td>205.3.4.93</td>
<td>Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>940</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>205.3.4.118</td>
<td>Modeling, Loads: 20Node Brick Surface Load with the Constant Pressure</td>
<td>970</td>
</tr>
<tr>
<td>205.3.4.119</td>
<td>Modeling, Loads: 20 Node Brick, Surface Load with Variable Pressure</td>
<td>971</td>
</tr>
<tr>
<td>205.3.4.120</td>
<td>Modeling, Loads: 27 Node Brick Surface Load with the Constant Pressure</td>
<td>972</td>
</tr>
<tr>
<td>205.3.4.121</td>
<td>Modeling, Loads: 27 Node Brick Surface Load with Variable Pressure</td>
<td>973</td>
</tr>
<tr>
<td>205.3.4.122</td>
<td>Modeling, Loads: Removing Loads</td>
<td>974</td>
</tr>
<tr>
<td>205.3.4.123</td>
<td>Modeling, Loads: Domain Reduction Method, DRM</td>
<td>975</td>
</tr>
<tr>
<td>205.3.4.124</td>
<td>Modeling, Wave Field for Creating DRM Loads: Add Wave Field</td>
<td>979</td>
</tr>
<tr>
<td>205.3.4.125</td>
<td>Modeling, Wave Field for Creating DRM Loads: Deconvolution</td>
<td>982</td>
</tr>
<tr>
<td>205.3.4.126</td>
<td>Modeling, Wave Field for Creating DRM Loads: Deconvolution to DRM</td>
<td>983</td>
</tr>
<tr>
<td>205.3.4.127</td>
<td>Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident SV Wave Potential Magnitude</td>
<td>984</td>
</tr>
<tr>
<td>205.3.4.128</td>
<td>Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident SV Wave Time Series Signal</td>
<td>987</td>
</tr>
<tr>
<td>205.3.4.129</td>
<td>Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident P Wave Potential Magnitude</td>
<td>990</td>
</tr>
<tr>
<td>205.3.4.130</td>
<td>Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident P Wave Time Series Signal</td>
<td>992</td>
</tr>
<tr>
<td>205.3.4.131</td>
<td>Modeling, Wave Field for Creating DRM Loads: DRM Inclined Motion</td>
<td>995</td>
</tr>
<tr>
<td>205.3.4.132</td>
<td>Modeling, Imposed Motions: through Loads, Motion Time History, Constant Time Step</td>
<td>996</td>
</tr>
<tr>
<td>205.3.4.133</td>
<td>Modeling, Imposed Motions: through Loads, Stochastic Motion Time History, Constant Time Step</td>
<td>997</td>
</tr>
<tr>
<td>205.3.4.134</td>
<td>Modeling, Imposed Motions: through Loads, Stochastic Random Process Motions, Constant Time Step</td>
<td>999</td>
</tr>
<tr>
<td>205.3.4.135</td>
<td>Modeling, Imposed Motions: through Loads, Motion Time History, Variable Time Step</td>
<td>1000</td>
</tr>
<tr>
<td>205.3.4.136</td>
<td>Modeling, Imposed Motions: Adding Load for Uniform Acceleration Time History</td>
<td>1001</td>
</tr>
<tr>
<td>205.3.4.137</td>
<td>Modeling, Imposed Motions: Remove Imposed Motions</td>
<td>1002</td>
</tr>
<tr>
<td>205.3.4.138</td>
<td>Modeling, Random Variable: Adding Gaussian Random Variables</td>
<td>1003</td>
</tr>
<tr>
<td>205.3.4.139</td>
<td>Modeling, Random Variable: Adding Gaussian Random Variables with Location</td>
<td>1004</td>
</tr>
<tr>
<td>205.3.4.140</td>
<td>Modeling, Random Variable: Adding Lognormal Random Variables</td>
<td>1005</td>
</tr>
</tbody>
</table>
205.3.4.159  Modeling, Random Field: Adding Random Field with Exponential Correlation  
1024

205.3.4.160  Modeling, Random Field: Adding Random Field with Triangular Correlation  
1025

205.3.4.161  Modeling, Random Field: Adding Random Field with Exponentially Damped  
Cosine Correlation . . . . . . . . . . . . . . . . . . . . . . 1026

205.3.4.162  Modeling, Random Field: Remove Random Fields . . . . . . . . . 1027

205.3.4.163  Modeling, Random Field: Adding Random Variable to Random Field.  . 1028

205.3.4.164  Modeling, Random Field: Remove Random Variable From Random Field 1029

205.3.4.165  Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève  
Expansion . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1030

205.3.4.166  Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève  
Expansion with Inverse Order. . . . . . . . . . . . . . . . . 1031

205.3.4.167  Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève  
Expansion Using HDF5 Input. . . . . . . . . . . . . . . . . 1032

205.3.4.168  Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève  
Expansion with Inverse Order Using HDF5 Input . . . . . . 1034

205.3.4.169  Modeling, Random Field: Output Hermite Polynomial Chaos Karhunen  
Loève Expansion Result . . . . . . . . . . . . . . . . . . . . 1036

205.3.4.170  Modeling, Random Field: Adding Random Field from Hermite Polynomial  
Chaos Karhunen Loève Expansion HDF5 File. . . . . . . . . . 1038

205.3.4.171  Modeling, Random Field: Adding Random Field from Marginal Distribu-  
tion and Correlation . . . . . . . . . . . . . . . . . . . . . . 1040

205.3.4.172  Modeling, Random Field: Add Triple Product of Hermite Polynomial  
Chaos Basis . . . . . . . . . . . . . . . . . . . . . . . . . . . 1041

205.3.4.173  Modeling, Random Field: Add Double Product of Hermite Polynomial  
Chaos Basis . . . . . . . . . . . . . . . . . . . . . . . . . . . 1042

205.3.4.174  Modeling, Random Field: Generate Triple Product of Hermite Polynomial  
Chaos Basis . . . . . . . . . . . . . . . . . . . . . . . . . . . 1043

205.3.4.175  Modeling, Random Field: Generate Double Product of Hermite Poly-  
nomial Chaos Basis . . . . . . . . . . . . . . . . . . . . . . 1045

205.3.4.176  Modeling, Random Field: Add Triple Product of Hermite Polynomial  
Chaos Basis Using HDF5 Input . . . . . . . . . . . . . . . . 1047
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>205.3.4.177</td>
<td>Modeling, Random Field: Add Double Product of Hermite Polynomial Chaos Basis Using HDF5 Input</td>
</tr>
<tr>
<td>205.3.4.178</td>
<td>Modeling, Solid-Fluid Interaction: Adding Solid-Fluid Interface</td>
</tr>
<tr>
<td>205.3.4.179</td>
<td>Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface, ESSI Element Nodes</td>
</tr>
<tr>
<td>205.3.4.180</td>
<td>Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface, ESSI Element Faces</td>
</tr>
<tr>
<td>205.3.4.181</td>
<td>Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface FOAM Nodes</td>
</tr>
<tr>
<td>205.3.4.182</td>
<td>Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface FOAM Faces</td>
</tr>
<tr>
<td>205.3.5</td>
<td>Simulation</td>
</tr>
<tr>
<td>205.3.5.1</td>
<td>Simulation, Solvers: Sequential Solvers</td>
</tr>
<tr>
<td>205.3.5.2</td>
<td>Simulation, Solvers: Parallel Solvers</td>
</tr>
<tr>
<td>205.3.5.3</td>
<td>Simulation: Static Solution Advancement</td>
</tr>
<tr>
<td>205.3.5.4</td>
<td>Simulation: Dynamic Solution Advancement with the Constant Time Step</td>
</tr>
<tr>
<td>205.3.5.5</td>
<td>Simulation: Dynamic Solution Advancement with Variable Time Step</td>
</tr>
<tr>
<td>205.3.5.6</td>
<td>Simulation: Generalized Eigenvalue Analysis</td>
</tr>
<tr>
<td>205.3.5.7</td>
<td>Simulation: Displacement Control</td>
</tr>
<tr>
<td>205.3.5.8</td>
<td>Simulation: Load, Control, Factor Increment</td>
</tr>
<tr>
<td>205.3.5.9</td>
<td>Simulation: Dynamic Integrator, Newmark Method</td>
</tr>
<tr>
<td>205.3.5.10</td>
<td>Simulation: Dynamic Integrator, Hilber Hughes Taylor, HHT, $\alpha$ Method</td>
</tr>
<tr>
<td>205.3.5.11</td>
<td>Simulation: Absolute Convergence Criteria</td>
</tr>
<tr>
<td>205.3.5.12</td>
<td>Simulation: Average Convergence Criteria</td>
</tr>
<tr>
<td>205.3.5.13</td>
<td>Simulation: Relative Convergence Criteria</td>
</tr>
<tr>
<td>205.3.5.14</td>
<td>Simulation: Solution Algorithms</td>
</tr>
<tr>
<td>205.3.5.15</td>
<td>Simulation: Constitutive Integration Algorithm</td>
</tr>
<tr>
<td>205.3.5.16</td>
<td>Simulation: Status Check</td>
</tr>
<tr>
<td>205.3.5.17</td>
<td>Simulation: Save State</td>
</tr>
<tr>
<td>205.3.5.18</td>
<td>Simulation: Restart Simulation</td>
</tr>
<tr>
<td>205.3.5.19</td>
<td>Simulation: Return Value for simulate Command</td>
</tr>
<tr>
<td>205.3.5.20</td>
<td>Simulation: New Elastic Loading Case</td>
</tr>
<tr>
<td>205.3.5.21</td>
<td>Simulation: Combine Elastic Load Cases</td>
</tr>
</tbody>
</table>
## Contents

205.3.5.22 Simulation, Dynamic Solution Advancement for Solid-Fluid Interaction ........................................ 1092
205.3.5.23 Simulation, Dynamic Solution Advancement for Stochastic Finite Element Method ......................................... 1093
205.3.5.24 Simulation, Sobol Sensitivity Analysis .................................................................................................... 1094
205.3.6 Output Options ........................................................................................................................................... 1098
205.3.6.1 Output Options: Enable/Disable Output ..................................................................................................... 1099
205.3.6.2 Output Options: Enable/Disable Element Output ...................................................................................... 1100
205.3.6.3 Output Options: Enable/Disable Displacement Output ................................................................................ 1101
205.3.6.4 Output Options: Enable/Disable Acceleration Output ............................................................................ 1102
205.3.6.5 Output Options: Enable/Disable Asynchronous Output .......................................................................... 1103
205.3.6.6 Output Options: Output Every n Steps .................................................................................................... 1104
205.3.6.7 Output Options: Output Support Reactions ............................................................................................. 1105
205.4 Checking the Model ...................................................................................................................................... 1106
205.5 Constitutive Testing ...................................................................................................................................... 1107
205.6 List of Available Commands (tentative, not up to date) ............................................................................... 1109
205.7 List of reserved keywords .......................................................................................................................... 1119
205.8 Integrated Development Environment (IDE) for DSL .................................................................................. 1132
205.9 Mesh Generation using GiD ......................................................................................................................... 1133
205.10 Model Development and Mesh Generation using gmesh ........................................................................ 1134
205.11 Model Input File Editing using Sublime .................................................................................................... 1135

## Output Formats

206.1 Chapter Summary and Highlights ................................................................................................................. 1137
206.2 Introduction ..................................................................................................................................................... 1137
206.3 Output Filename and Format ........................................................................................................................ 1137
206.3.1 Sequential .................................................................................................................................................... 1137
206.3.2 Parallel ........................................................................................................................................................ 1138
206.4 Output Units ................................................................................................................................................... 1139
206.5 Data organization ........................................................................................................................................... 1139
206.5.1 The Root group ......................................................................................................................................... 1139
206.5.2 The Model group .................................................................................................................................. 1143
206.5.3 The Nodes group .................................................................................................................................. 1143
206.5.3.1 Number_of_DOFs ............................................................................................................................. 1145
206.5.3.2 Partition ............................................................................................................................................. 1145
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>206.8.2</td>
<td>Brick Elements</td>
<td>1169</td>
</tr>
<tr>
<td>206.8.3</td>
<td>ShearBeam</td>
<td>1169</td>
</tr>
<tr>
<td>206.8.4</td>
<td>ElasticBeam</td>
<td>1170</td>
</tr>
<tr>
<td>206.8.5</td>
<td>4NodeShell_ANDES</td>
<td>1170</td>
</tr>
<tr>
<td>206.8.6</td>
<td>BeamColumnDispFiber3d</td>
<td>1170</td>
</tr>
<tr>
<td>206.8.7</td>
<td>Contact/Interface Elements</td>
<td>1171</td>
</tr>
<tr>
<td>206.9</td>
<td>Energy output format</td>
<td>1172</td>
</tr>
<tr>
<td>206.9.1</td>
<td>Input Energy</td>
<td>1172</td>
</tr>
<tr>
<td>206.9.2</td>
<td>Energy Density Quantity at Gauss Point</td>
<td>1172</td>
</tr>
<tr>
<td>206.9.3</td>
<td>Average Energy Density Quantity of Element</td>
<td>1173</td>
</tr>
<tr>
<td>206.9.4</td>
<td>Energy Quantity of Element</td>
<td>1174</td>
</tr>
</tbody>
</table>

207 Real-ESSI Pre Processing and Model Development Methods


207.1 Introduction 1177
207.2 Model Development Using gmESSI 1177
207.2.1 Introduction to gmESSI 1177
207.2.1.1 Getting Started 1177
207.2.1.2 Running gmESSI 1179
207.2.2 Gmsh Physical Groups and Geometrical Entities 1184
207.2.2.1 Geometrical Entities 1184
207.2.2.2 Physical Groups 1184
207.2.3 gmESSI Command Description 1190
207.2.3.1 gmESSI Syntax 1190
207.2.3.2 gmESSI Command’s Physical Group 1193
207.2.4 gmESSI Output 1194
207.2.4.1 Directory Example_2_ESSI_Simulation 1194
207.2.4.2 Translation Log Terminal 1195
207.2.4.3 Element File (element.fei) 1198
207.2.4.4 Node File (node.fei) 1198
207.2.4.5 Load File (load.fei) 1198
207.2.4.6 Analysis File (main.fei) 1199
207.2.4.7 Mesh File (XYZ.msh) 1200
207.2.4.8 Updated ESSI Tags Terminal 1200
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>207.2.5</td>
<td>gmESSI Commands</td>
</tr>
<tr>
<td>207.2.5.1</td>
<td>Singular Commands</td>
</tr>
<tr>
<td>207.2.5.2</td>
<td>Add Node Commands</td>
</tr>
<tr>
<td>207.2.5.3</td>
<td>Nodal Commands: Operates On All Nodes of the defined Physical Group</td>
</tr>
<tr>
<td>207.2.5.4</td>
<td>General Elemental Commands: Operates On All Elements of the defined Physical Group</td>
</tr>
<tr>
<td>207.2.5.5</td>
<td>Elemental Commands: Operates On All Elements of the defined Physical Group</td>
</tr>
<tr>
<td>207.2.5.6</td>
<td>Elemental Compound Commands: Operates On All Surface Elements of the defined Physical Group [Surface Loads]</td>
</tr>
<tr>
<td>207.2.5.7</td>
<td>Special Commands</td>
</tr>
<tr>
<td>207.2.5.8</td>
<td>Connect Command</td>
</tr>
<tr>
<td>207.2.5.9</td>
<td>Write Command</td>
</tr>
<tr>
<td>207.2.5.10</td>
<td>Write DRM HDF5 Command</td>
</tr>
<tr>
<td>207.2.6</td>
<td>Steps For Using gmESSI tool</td>
</tr>
<tr>
<td>207.2.6.1</td>
<td>Building geometry (.geo) file in Gmsh</td>
</tr>
<tr>
<td>207.2.6.2</td>
<td>Generate mesh (.msh) file in Gmsh</td>
</tr>
<tr>
<td>207.2.6.3</td>
<td>Writing all gmESSI Commands for the model</td>
</tr>
<tr>
<td>207.2.6.4</td>
<td>Executing gmESSI on Example 1.gmessi input file</td>
</tr>
<tr>
<td>207.2.6.5</td>
<td>Running Real-ESSI and visualization in paraview</td>
</tr>
<tr>
<td>207.2.7</td>
<td>Illustrative Examples</td>
</tr>
<tr>
<td>207.2.7.1</td>
<td>Modeling of Cantilever Beam With Surface Load [Example_2]</td>
</tr>
<tr>
<td>207.2.7.2</td>
<td>Modeling of a embedded shells and beam in Solids [Example_5]</td>
</tr>
<tr>
<td>207.2.8</td>
<td>Realistic Models Developed Using gmESSI</td>
</tr>
<tr>
<td>207.3</td>
<td>Introduction to SASSI-ESSI Translator</td>
</tr>
</tbody>
</table>

208 Real-ESSI Post Processing Methods

<p>| 208.1 | Introduction |
| 208.2 | Model Results Post-Processing |
| 208.3 | Time Histories Plotting |
| 208.4 | Post Processing and Visualization using ParaView |
| 208.4.1 | Visualization in ParaView: Features |
| 208.4.1.1 | PVESSEISReader Visualization Options |
| 208.4.1.2 | Sequential Visualization |</p>
<table>
<thead>
<tr>
<th>208.4.1.3</th>
<th>Remote Visualization</th>
<th>1253</th>
</tr>
</thead>
<tbody>
<tr>
<td>208.4.1.4</td>
<td>Parallel Visualization</td>
<td>1253</td>
</tr>
<tr>
<td>208.4.1.5</td>
<td>General Field Visualization</td>
<td>1255</td>
</tr>
<tr>
<td>208.4.1.6</td>
<td>Relative Displacement Visualization</td>
<td>1262</td>
</tr>
<tr>
<td>208.4.1.7</td>
<td>Visualizing Element’s Partition</td>
<td>1264</td>
</tr>
<tr>
<td>208.4.1.8</td>
<td>Gauss Mesh Visualization Options</td>
<td>1265</td>
</tr>
<tr>
<td>208.4.1.9</td>
<td>Gauss To Node Interpolation Mode Visualization</td>
<td>1268</td>
</tr>
<tr>
<td>208.4.1.10</td>
<td>upU Visualization</td>
<td>1270</td>
</tr>
<tr>
<td>208.4.1.11</td>
<td>Eigen Mode Visualization</td>
<td>1272</td>
</tr>
<tr>
<td>208.4.1.12</td>
<td>Visualizing Physical Node and Element groups</td>
<td>1273</td>
</tr>
<tr>
<td>208.4.1.13</td>
<td>Using Threshold to Visualize Certain Elements</td>
<td>1277</td>
</tr>
</tbody>
</table>

209 Software Platform Build Process


209.1 Chapter Summary and Highlights                                | 1280 |
209.2 Introduction to the Real-ESSI Simulator Program              | 1281 |
209.3 Real-ESSI Program Executables Download and Install           | 1281 |
209.3.1 Sequential Version of Real-ESSI Program                    | 1281 |
209.3.2 Parallel Version of Real-ESSI Program                      | 1281 |
209.3.3 Real-ESSI Executable Downloads                             | 1282 |
209.4 Real-ESSI Simulator System Install                           | 1282 |
209.5 Quick Build Procedures for Sequential and Parallel Versions of the Real-ESSI Program | 1282 |
209.5.1 Update Compiler for Ubuntu 16.04, while 18.04 already has the latest version | 1282 |
209.5.2 Sequential Real-ESSI Build                                 | 1283 |
209.5.2.1 Libraries Required for Building Dependencies for Ubuntu 16.04 and 18.04 | 1283 |
209.5.2.2 Libraries Required for Building Utilities for Ubuntu 16.04 and 18.04 | 1284 |
209.5.2.3 Obtain Real-ESSI Sources                                 | 1284 |
209.5.2.4 Compile Real-ESSI Dependency Libraries                  | 1286 |
209.5.2.5 Compile and link Real-ESSI Program                      | 1286 |
209.5.3 Parallel Real-ESSI Build                                  | 1287 |
209.5.3.1 Libraries Required for Building Dependencies for Ubuntu 16.04 and 18.04 | 1287 |
209.5.3.2 Libraries Required for Building Utilities for Ubuntu 16.04 and 18.04 | 1287 |
209.5.3.3 Obtain Real-ESSI Sources                                 | 1288 |
209.5.3.4 Compile Real-ESSI Dependency Libraries                  | 1289 |
209.12.2 Copy an Existing Image .......................................................... 1320
209.12.3 Create a New Image ............................................................... 1321
209.12.4 Build AWS ESSI Image from Scratch ................................. 1323
209.12.5 Update an Existing Image ...................................................... 1327
209.12.6 Upload an Existing Real-ESSI Simulator Image to AWS MarketPlace .... 1328

210 Software Platform Procurement, Distribution

(2019-2020-2021-) .................................................................................. 1329
210.1 Chapter Summary and Highlights .................................................. 1330
210.2 Real-ESSI Program Executables Download and Install .................. 1330
  210.2.1 Sequential Version of Real-ESSI Program. ............................... 1330
  210.2.2 Parallel Version of Real-ESSI Program. ................................. 1330
  210.2.3 Real-ESSI Executable Downloads. ............................................ 1330
210.3 Real-ESSI Simulator Install as Container through Docker ............... 1331
  210.3.1 Real-ESSI Docker Image Development .................................... 1331
  210.3.2 Running Real-ESSI Container through Docker ....................... 1334
  210.3.3 Performance of Real-ESSI Container ...................................... 1336
210.4 Real-ESSI Simulator System Install ............................................. 1336
  210.4.1 Student Manual for Real-ESSI Simulator System Install .......... 1336

211 Software Platform Cloud Computing

211.1 Chapter Summary and Highlights .................................................. 1339
211.2 Real-ESSI Cloud Computing Overview. ....................................... 1339
  211.2.1 Real-ESSI Cloud Service Content .......................................... 1339
211.3 Launch Real-ESSI Instance on AWS ........................................... 1341
  211.3.1 Launch Real-ESSI Instance from AWS Private Images .......... 1341
  211.3.2 Launch Real-ESSI Instance from AWS Market Place ............. 1353
211.4 Connect to Real-ESSI Instance on AWS ....................................... 1353
  211.4.1 Install X2GO Client ............................................................... 1353
    211.4.1.1 Installing X2GO client on Ubuntu Linux ......................... 1353
    211.4.1.2 Installing X2GO client on Apple Mac ............................ 1354
    211.4.1.3 Installing X2GO client on Windows ............................... 1354
    211.4.1.4 Installing X2GO client on other operating systems ......... 1354
  211.4.2 Configure the Client-Side of X2GO ......................................... 1355
  211.4.3 Connect to the Launched Instance ......................................... 1356
211.5 Terminate Real-ESSI Instance on AWS ......................................................... 1356
211.6 Adding Permission for Private Real-ESSI Image to User AWS Accounts .......... 1357
211.7 Real-ESSI Instructional Videos Cloud Computing ........................................ 1358
  211.7.0.1 Installing X2GO for Windows ...................................................... 1358
  211.7.0.2 Installing X2GO for Macintosh ..................................................... 1358
  211.7.0.3 Installing X2GO for Linux .......................................................... 1358
  211.7.0.4 Launch AWS Marketplace .......................................................... 1358
  211.7.0.5 Access Running Instance on AWS ............................................... 1358
  211.7.0.6 Start Real-ESSI Program on AWS .............................................. 1358
  211.7.0.7 Run Real-ESSI Example Model on AWS ...................................... 1358
  211.7.0.8 Visualize Real-ESSI Example Model on AWS .............................. 1358
  211.7.0.9 Post-Process, Visualize Real-ESSI Results on AWS ...................... 1358
211.8 Cost of AWS EC2 .................................................................................. 1359
  211.8.1 Cost of Running Real-ESSI on AWS .............................................. 1361
    211.8.1.1 Small Size Real-ESSI Example ............................................... 1361
    211.8.1.2 Medium Size Real-ESSI Example ........................................... 1367
    211.8.1.3 Large Example ................................................................. 1370
  211.8.2 AWS for Education ....................................................................... 1374
  211.8.3 AWS for Government .................................................................... 1375
  211.8.3.1 AWS GovCloud .................................................................. 1375
  211.8.3.2 AWS Secret Region ............................................................. 1375
  212.1 Chapter Summary and Highlights ....................................................... 1377
  212.2 Introduction ..................................................................................... 1377
  212.3 The NRC ESSI Computer .................................................................. 1377
    212.3.1 Version: December 2010 ........................................................... 1377
    212.3.2 Version: April 2012 ................................................................. 1377
300 Verification and Validation ........................................................................... 1379
    301.1 Chapter Summary and Highlights ...................................................... 1381
301.2 Verification and Validation

301.2.1 Trusting Simulation Tools

301.2.2 Importance of V & V

301.2.3 Maturity of Computational Simulations

301.2.4 Role of Verification and Validation

301.2.4.1 Alternative V & V Definitions

301.2.4.2 Certification and Accreditation

301.2.4.3 Independence of Computational Confidence Assessment.

301.2.5 Detailed Look at Verification and Validation

301.3 Prediction

301.3.1 Relation Between Validation and Prediction

301.4 Application Domain.

301.4.1 Importance of Models and Numerical Simulations

301.4.2 Prediction under Uncertainty

302 Source Code Verification

302.1 Chapter Summary and Highlights

302.1.1 Numerical Algorithm Verification

302.1.2 Software Quality Assurance.

303 Code Stability Verification

303.1 Chapter Summary and Highlights

303.2 Introduction to Code Stability

303.3 Motivation

303.4 The framework of the automatic test

303.5 Installation and Tutorial

303.5.1 Installation

303.5.2 Tutorial

303.5.2.1 Run all verification test cases

303.5.2.2 Run a single type of verification test cases
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>303.6</td>
<td>The underlying implementation of the automatic test</td>
<td>1398</td>
</tr>
<tr>
<td>303.6.1</td>
<td>Generate the original results</td>
<td>1398</td>
</tr>
<tr>
<td>303.6.2</td>
<td>Run essi and make comparison</td>
<td>1399</td>
</tr>
<tr>
<td>303.6.2.1</td>
<td>The terminal output/log comparison</td>
<td>1399</td>
</tr>
<tr>
<td>303.6.2.2</td>
<td>Reduce the comparison items and comparison time</td>
<td>1400</td>
</tr>
<tr>
<td>303.7</td>
<td>Report Sample</td>
<td>1400</td>
</tr>
<tr>
<td>303.7.1</td>
<td>Passed test case</td>
<td>1400</td>
</tr>
<tr>
<td>303.7.2</td>
<td>Failed test case</td>
<td>1400</td>
</tr>
<tr>
<td>303.7.3</td>
<td>Version information</td>
<td>1401</td>
</tr>
<tr>
<td>303.7.4</td>
<td>Statistics</td>
<td>1401</td>
</tr>
<tr>
<td>303.8</td>
<td>Future contribution</td>
<td>1401</td>
</tr>
<tr>
<td>304</td>
<td>Verification and Validation for Constitutive Problems</td>
<td>1402</td>
</tr>
<tr>
<td>304.1</td>
<td>Chapter Summary and Highlights</td>
<td>1403</td>
</tr>
<tr>
<td>304.2</td>
<td>Verification of Constitutive Integration</td>
<td>1403</td>
</tr>
<tr>
<td>304.2.1</td>
<td>Error Assessment</td>
<td>1403</td>
</tr>
<tr>
<td>304.2.2</td>
<td>Constitutive Level Convergence</td>
<td>1408</td>
</tr>
<tr>
<td>304.3</td>
<td>Validation of Constitutive Model Predictions</td>
<td>1412</td>
</tr>
<tr>
<td>304.3.1</td>
<td>Dafalias Manzari Material Model</td>
<td>1412</td>
</tr>
<tr>
<td>305</td>
<td>Verification and Validation for Static and Dynamic Finite Element Level Solution Advancement Algorithms</td>
<td>1417</td>
</tr>
<tr>
<td>305.1</td>
<td>Chapter Summary and Highlights</td>
<td>1418</td>
</tr>
<tr>
<td>305.2</td>
<td>Verification for Static Solution Advancement</td>
<td>1418</td>
</tr>
<tr>
<td>305.3</td>
<td>Verification for Dynamic Solution Advancement</td>
<td>1418</td>
</tr>
<tr>
<td>305.3.1</td>
<td>Verification for Dynamic Solution Advancement, Newmark Method</td>
<td>1418</td>
</tr>
<tr>
<td>305.3.2</td>
<td>Verification Example Description.</td>
<td>1420</td>
</tr>
<tr>
<td>305.3.3</td>
<td>Verification for Dynamic Solution Advancement, Hilber-Hughes-Taylor Method</td>
<td>1427</td>
</tr>
<tr>
<td>305.3.3.1</td>
<td>Verification Results for Hilber Hughes Taylor (HHT) Solution Advancement Algorithm</td>
<td>1428</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>306 Verification and Validation for Static and Dynamic Behavior of Single Phase, Solid Elements</td>
<td>1434</td>
<td></td>
</tr>
<tr>
<td>306.1 Chapter Summary and Highlights</td>
<td>1435</td>
<td></td>
</tr>
<tr>
<td>306.2 Verification of Static, Single Phase Solid Modeling and Simulation</td>
<td>1435</td>
<td></td>
</tr>
<tr>
<td>306.2.1 Beam theory</td>
<td>1435</td>
<td></td>
</tr>
<tr>
<td>306.2.2 Verification of 8 node brick cantilever beam (static)</td>
<td>1436</td>
<td></td>
</tr>
<tr>
<td>306.2.3 Verification of 27 node brick cantilever beam (static)</td>
<td>1438</td>
<td></td>
</tr>
<tr>
<td>306.2.4 Verification of 8NodeBrick cantilever beams</td>
<td>1440</td>
<td></td>
</tr>
<tr>
<td>306.2.5 Verification of 8NodeBrick cantilever beam for different Poisson’s ratio</td>
<td>1442</td>
<td></td>
</tr>
<tr>
<td>306.2.6 Test of irregular shaped 8NodeBrick cantilever beams.</td>
<td>1447</td>
<td></td>
</tr>
<tr>
<td>306.2.7 Verification of 8NodeBrick stress in cantilever beams.</td>
<td>1455</td>
<td></td>
</tr>
<tr>
<td>306.2.8 Verification of 8NodeBrick square plate with four edges clamped</td>
<td>1458</td>
<td></td>
</tr>
<tr>
<td>306.2.9 Verification of 8NodeBrick square plate with four edges simply supported</td>
<td>1463</td>
<td></td>
</tr>
<tr>
<td>306.2.10 Verification of 8NodeBrick circular plate with all edges clamped</td>
<td>1468</td>
<td></td>
</tr>
<tr>
<td>306.2.11 Verification of 8NodeBrick circular plate with all edges simply supported</td>
<td>1473</td>
<td></td>
</tr>
<tr>
<td>306.2.12 Verification of 8NodeBrick Finite Element for Boussinesq Problem</td>
<td>1478</td>
<td></td>
</tr>
<tr>
<td>306.2.12.1 Introduction</td>
<td>1478</td>
<td></td>
</tr>
<tr>
<td>306.2.12.2 Description of the Verification Model</td>
<td>1478</td>
<td></td>
</tr>
<tr>
<td>306.2.12.3 Results</td>
<td>1479</td>
<td></td>
</tr>
<tr>
<td>306.2.12.4 Error Analysis</td>
<td>1480</td>
<td></td>
</tr>
<tr>
<td>306.2.13 Verification of 8NodeBrick Finite Element for Collapsed Brick Shapes</td>
<td>1482</td>
<td></td>
</tr>
<tr>
<td>306.2.13.1 Test procedure</td>
<td>1482</td>
<td></td>
</tr>
<tr>
<td>306.2.13.2 Test result</td>
<td>1483</td>
<td></td>
</tr>
<tr>
<td>306.2.14 Verification of 27 node brick cantilever beam (static)</td>
<td>1486</td>
<td></td>
</tr>
<tr>
<td>306.2.15 Verification of 27NodeBrick cantilever beam for different Poisson’s ratio</td>
<td>1489</td>
<td></td>
</tr>
<tr>
<td>306.2.16 Test of irregular shaped 27NodeBrick cantilever beams</td>
<td>1496</td>
<td></td>
</tr>
<tr>
<td>306.2.17 Verification of 27NodeBrick stress in cantilever beams</td>
<td>1504</td>
<td></td>
</tr>
<tr>
<td>306.2.18 Verification of 27NodeBrick square plate with four edges clamped</td>
<td>1507</td>
<td></td>
</tr>
<tr>
<td>306.2.19 Verification of 27NodeBrick square plate with four edges simply supported</td>
<td>1512</td>
<td></td>
</tr>
<tr>
<td>306.2.20 Verification of 27NodeBrick circular plate with all edges clamped</td>
<td>1518</td>
<td></td>
</tr>
<tr>
<td>306.2.21 Verification of 27NodeBrick circular plate with all edges simply supported</td>
<td>1523</td>
<td></td>
</tr>
<tr>
<td>306.2.22 Verification of 27NodeBrick Finite Element for Boussinesq Problem</td>
<td>1528</td>
<td></td>
</tr>
<tr>
<td>306.2.22.1 Introduction</td>
<td>1528</td>
<td></td>
</tr>
</tbody>
</table>
306.2.22.2 Description of the Verification Model .................................................... 1528
306.2.22.3 Results ..................................................................................................... 1529
306.2.22.4 Error Analysis .......................................................................................... 1530
306.3 Verification of Dynamic, Single Phase Solid Modeling and Simulation ............... 1531
307 Verification and Validation for Static and Dynamic Behavior of Structural Elements


307.1 Chapter Summary and Highlights ................................................................. 1532
307.2 Verification of Static, Beam-Column Finite Element Modeling and Simulation. 1533
307.3 Bernoulli Beam Elements with 12DOFs and 9DOFs. ................................. 1533
  307.3.1 FEM Model ............................................................................................... 1533
  307.3.2 Static Analysis ........................................................................................... 1533
  307.3.3 Dynamic Analysis ...................................................................................... 1534
  307.3.4 Bernoulli Beam, Comparison of Eigen Frequencies ................................. 1536
307.4 Timoshenko Beam ......................................................................................... 1537
307.5 Verification of Shell (Felippa-ANDES) Finite Element Modeling and Simulation 1541
  307.5.1 Static Tests ................................................................................................ 1541
    307.5.1.1 Bending Component Verification ......................................................... 1541
    307.5.1.2 Membrane Component Verification ................................................ 1546
  307.5.2 Dynamic Tests .......................................................................................... 1550
    307.5.2.1 Bending Component ............................................................................ 1550
    307.5.2.2 Membrane Component .................................................................... 1551
    307.5.2.3 Geometric Transformations ............................................................... 1552
307.6 Verification of 4NodeANDES elements ....................................................... 1553
  307.6.1 Verification of 4NodeANDES cantilever beams ..................................... 1553
  307.6.2 Verification of 4NodeANDES cantilever beam for different Poisson’s ratio . 1557
  307.6.3 Test of irregular shaped 4NodeANDES cantilever beams. .................... 1569
  307.6.4 Verification of 4NodeANDES square plate with four edges clamped ....... 1580
  307.6.5 Verification of 4NodeANDES square plate with four edges simply supported 1586
  307.6.6 Verification of 4NodeANDES circular plate with all edges clamped ......... 1592
  307.6.7 Verification of 4NodeANDES circular plate with all edges simply supported 1597
308 Verification and Validation for Static and Dynamic Behavior of Special Elements
(Contacts/Interfaces/Joints, Gap/Frictional, Isolators)


308.1 Chapter Summary and Highlights .................................................................................................................. 1602

308.2 Verification of Static Penalty Contact/Interface/Joint Element Modeling and Simulation ............................................... 1603

308.2.1 Static Normal Contact/Interface/Joint Verification ......................................................................................... 1603

308.2.1.1 Case 1: Monotonic Loading with initial gap $\delta_{in} = 0.1m$ .......................................................... 1604

308.2.1.2 Case 2: Monotonic Loading with no initial gap $\delta_{in} = 0m$ ....................................................... 1604

308.2.1.3 Case 3: Cyclic Loading with initial gap $\delta_{in} = 0.1m$ ................................................................. 1604

308.2.1.4 Case 4: Cyclic Loading with no initial gap $\delta_{in} = 0m$ ............................................................. 1604

308.2.2 Static Frictional Tangential Contact/Interface/Joint Verification ................................................................. 1610

308.2.2.1 Case 1: Verification of the yield surface for different loading angles with fixed normal confinement..... 1611

308.3 Verification of Static and Dynamic Contact/Interface/Joint Element Modeling and Simulation ........................................ 1614

308.3.1 Truss Examples ........................................................................................................................................ 1615

308.3.2 Single Brick Element Examples ............................................................................................................... 1617

308.3.3 Double Brick Element Examples ............................................................................................................ 1617

308.4 Verification of Static and Dynamic Coupled (Saturated) Contact/Interface/Joint Element Modeling and Simulation .................................................. 1623

308.4.1 Dry u-p-U Contact/Interface/Joint to the Ground ....................................................................................... 1623

308.4.2 u-p-U Contact/Interface/Joint to the ground ......................................................................................... 1625

308.5 Verification of Static, Isolator Element Modeling and Simulation ....................................................................... 1629

309 Verification and Validation for Coupled, Porous Solid – Pore Fluid Problems


309.1 Chapter Summary and Highlights .................................................................................................................. 1630

309.2 Introduction .................................................................................................................................................. 1631

309.3 Drilling of a well ........................................................................................................................................... 1631

309.3.1 The Problem ........................................................................................................................................... 1631

309.3.2 Analytical Solution ................................................................................................................................... 1632

309.3.3 Discussion of the Results ....................................................................................................................... 1633

309.4 The Case of a Spherical Cavity ..................................................................................................................... 1637

309.4.1 The Problem ........................................................................................................................................... 1637
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>309.4.2 Analytical Solution</td>
<td>1638</td>
</tr>
<tr>
<td>309.4.3 Discussion of the Results</td>
<td>1639</td>
</tr>
<tr>
<td>309.5 Line Injection of a fluid in a Reservoir</td>
<td>1640</td>
</tr>
<tr>
<td>309.5.1 The Problem</td>
<td>1640</td>
</tr>
<tr>
<td>309.5.2 Analytical Solution</td>
<td>1641</td>
</tr>
<tr>
<td>309.5.3 Discussion of the Results</td>
<td>1643</td>
</tr>
<tr>
<td>309.6 Shock Wave Propagation in Saturated Porous Medium</td>
<td>1648</td>
</tr>
<tr>
<td>309.7 Vertical Consolidation of a soil layer by Coussy (2004)</td>
<td>1649</td>
</tr>
<tr>
<td>309.7.1 Brief review of Analytical Solution for Consolidation by Coussy (2004)</td>
<td>1649</td>
</tr>
<tr>
<td>309.7.2 Numerical Analysis</td>
<td>1652</td>
</tr>
<tr>
<td>309.7.3 Discussion of Numerical Results - Conclusions</td>
<td>1654</td>
</tr>
<tr>
<td>309.8 One dimensional wave propagation in elastic porous media subjected to step displacement boundary condition</td>
<td>1656</td>
</tr>
<tr>
<td>309.8.1 Brief review of Analytical Solution by Gajo and Mongiovi (1995)</td>
<td>1656</td>
</tr>
<tr>
<td>309.8.2 Numerical Analysis</td>
<td>1666</td>
</tr>
<tr>
<td>309.8.3 Discussion of Numerical Results - Conclusions</td>
<td>1696</td>
</tr>
<tr>
<td>309.9 One dimensional wave propagation in elastic porous media subjected to step loading at the surface</td>
<td>1696</td>
</tr>
<tr>
<td>309.9.1 Brief review of Analytical Solution by de Boer et al. (1993)</td>
<td>1696</td>
</tr>
<tr>
<td>309.9.2 Numerical Analysis</td>
<td>1697</td>
</tr>
<tr>
<td>309.9.3 Discussion of Numerical Results - Conclusions</td>
<td>1699</td>
</tr>
<tr>
<td>309.10 One dimensional wave propagation in elastic porous media subjected to step velocity boundary condition</td>
<td>1701</td>
</tr>
<tr>
<td>309.10.1 Brief review of Analytical Solution by Hiremath et al. (1988)</td>
<td>1701</td>
</tr>
<tr>
<td>309.10.2 Numerical Analysis</td>
<td>1703</td>
</tr>
<tr>
<td>309.10.3 Discussion of Numerical Results - Conclusions</td>
<td>1705</td>
</tr>
<tr>
<td>310 Verification and Validation for Seismic Wave Propagation Problems</td>
<td></td>
</tr>
<tr>
<td>310.1 Chapter Summary and Highlights</td>
<td>1721</td>
</tr>
<tr>
<td>310.2 Wavelet Seismic Signals</td>
<td>1721</td>
</tr>
<tr>
<td>310.2.1 Ricker Wavelet</td>
<td>1721</td>
</tr>
<tr>
<td>310.2.2 Ormsby Wavelet</td>
<td>1722</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>310.3</td>
<td>Finite Element Mesh Size Effects on Seismic Wave Propagation Modeling and Simulation</td>
</tr>
<tr>
<td>310.3.1</td>
<td>Analysis Cases</td>
</tr>
<tr>
<td>310.3.2</td>
<td>Comparison of Case 1 and 2</td>
</tr>
<tr>
<td>310.3.3</td>
<td>Comparison of Case 3 and 4</td>
</tr>
<tr>
<td>310.3.4</td>
<td>Comparison of Cases 3, 4, and 6.</td>
</tr>
<tr>
<td>310.3.5</td>
<td>Comparison of Case 7 and 8</td>
</tr>
<tr>
<td>310.3.6</td>
<td>Comparison of Case 9, 10, and 11</td>
</tr>
<tr>
<td>310.3.7</td>
<td>Comparison of Case 12, 13, and 14.</td>
</tr>
<tr>
<td>310.4</td>
<td>Damping of the Outgoing Waves</td>
</tr>
<tr>
<td>310.4.1</td>
<td>Comparison of Rayleigh Damping and Caughey 4th Order Damping</td>
</tr>
<tr>
<td>310.4.2</td>
<td>Parametric Study on Effect of Rayleigh Damping on Reflected Waves.</td>
</tr>
<tr>
<td>310.5</td>
<td>Mesh Size Effects for Linear (8 Node Brick) and Quadratic (27 Node Brick) Finite Elements on Wave Propagation</td>
</tr>
<tr>
<td>310.6</td>
<td>Verification of the Seismic Input (Domain Reduction Method) for 3C, Inclined Seismic Wave Fields</td>
</tr>
<tr>
<td>310.6.1</td>
<td>Inclined, 3C Seismic Waves in a Free Field</td>
</tr>
<tr>
<td>310.6.1.1</td>
<td>Ricker Wavelets</td>
</tr>
<tr>
<td>310.6.2</td>
<td>Vertical (1C) Seismic Waves in a Free Field</td>
</tr>
<tr>
<td>310.6.2.1</td>
<td>Morgan Hill and Kocaeli Earthquakes.</td>
</tr>
<tr>
<td>310.6.3</td>
<td>Earthquake-Soil-Structure Interaction Verification for Simulated Northridge Seismic Motions</td>
</tr>
<tr>
<td>310.6.4</td>
<td>Curious Case of 1C versus 3C modeling</td>
</tr>
<tr>
<td>310.6.5</td>
<td>Earthquake-Soil-Structure Interaction for Surface and Embedded Structures</td>
</tr>
<tr>
<td>310.6.5.1</td>
<td>Uniform half-space</td>
</tr>
<tr>
<td>310.6.5.2</td>
<td>Layered half-space</td>
</tr>
<tr>
<td>310.6.5.3</td>
<td>Layered Layered over rigid lower boundary</td>
</tr>
<tr>
<td>310.7</td>
<td>Case History: Simple Structure on Nonlinear Soil</td>
</tr>
<tr>
<td>310.7.1</td>
<td>Simplified Models for Verification</td>
</tr>
<tr>
<td>310.7.1.1</td>
<td>Model Description</td>
</tr>
<tr>
<td>310.7.1.2</td>
<td>Static Pushover Test on Elastic-Plastic Soil Column</td>
</tr>
<tr>
<td>310.7.1.3</td>
<td>Dynamic Test on Elastic Soil Column.</td>
</tr>
<tr>
<td>310.7.1.4</td>
<td>Dynamic Test on Elastic-Plastic Soil Column.</td>
</tr>
<tr>
<td>310.7.1.5</td>
<td>2d Model</td>
</tr>
</tbody>
</table>
312.2.2 Dam Break .................................................... 1821
312.3 Verification & Validation ...................................... 1821
312.3.1 Free Surface Flow validation ............................. 1821
312.3.2 Mass conservation verification ......................... 1823
312.3.3 SFI Verification & Validation ............................ 1823

313 Quality Management System
313.1 Chapter Summary and Highlights .......................... 1827
313.2 Reasoning Behind this Activity ............................ 1827
313.3 Real-ESSI Simulator System Quality Management System, based on ISO/IEC/IEEE 90003 Standard ............................ 1828
313.3.1 Real-ESSI Simulator Developer Organization ............ 1828
313.3.1.1 Internal Issues .................................... 1828
313.3.1.2 Internal Issues .................................... 1828
313.3.1.3 External and Internal Issues for the Real-ESSI Simulator ............................ 1829
313.3.1.4 Needs and expectations of Interested Parties ........... 1829
313.3.2 Scope of the Real-ESSI Simulator Quality Management System ............................ 1830
313.4 Real-ESSI Simulator System Quality Management System, based on ASME NQA-1 Standard ............................ 1831
313.4.0.1 ASME NQA-1 for the Real-ESSI Simulator System ............................ 1831

314 Comparison with Other Programs
                              (2016-2017-2020-2021-) .......................... 1832
314.1 Chapter Summary and Highlights .......................... 1833
314.1.1 Reasoning Behind this Activity ............................ 1833

400 Education, Training and Modeling, Simulation Examples 1834

401 Ten Week Course on Nonlinear Finite Element Methods for Realistic Modeling and Simulation of Earthquakes, and Soils, and Structures, and their Interaction, Real-ESSI
                              (1998-2021-) .......................... 1835
401.1 Delivery .................................................... 1836
401.2 Objectives .................................................... 1836
401.3 Additional Information ..................................... 1837
403 Online Education and Training

(2019-2020-2021-)

403.1 Real-ESSI Simulator Online Education and Training

403.1.1 Modeling and Simulations for ESSI.

403.1.1.1 Introduction to Modeling and Simulation

403.1.1.2 Introduction to Modeling Simplifications, Epistemic Uncertainty

403.1.1.3 Introduction to Parametric, Aleatory Uncertainty

403.1.2 Real-ESSI Simulator Modeling and Simulation System

403.1.3 Finite Element Method

403.1.3.1 Background

403.1.3.2 Nonlinear, Inelastic FEM

403.1.3.3 Dynamic FEM

403.1.4 Deterministic Elasto-Plasticity

403.1.4.1 Theory Background

403.1.4.2 Elastic-Plastic Material Model Choices

403.1.4.3 Calibrating Elastic-Plastic Material Models

403.1.5 Seismic Energy Dissipation

403.1.5.1 Theory Background

403.1.5.2 Illustrative Examples

403.1.6 Probabilistic Elasto-Plasticity and Stochastic Elastic-Plastic Finite Element Method

403.1.6.1 Theory Background

403.1.6.2 Choice and Calibration of Probabilistic Material Models and Probabilistic Seismic Loads

403.1.6.3 Simple Probabilistic Examples

403.1.6.4 Probabilistic Wave Propagation Examples

403.1.7 Seismic Motions

403.1.8 Earthquake Soil Structure Interaction

403.1.9 Verification and Validation

403.1.10 High Performance Computing

403.1.10.1 HPC Introduction

403.1.10.2 HPC and Real-ESSI

403.1.10.3 Real-ESSI Parallel Computing Examples

403.1.11 Real-ESSI Simulator Examples
## Constitutive, Material Behaviour Examples

*(2016-2017-2019-)*

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>404.1</td>
<td>Chapter Summary and Highlights</td>
<td>1881</td>
</tr>
<tr>
<td>404.2</td>
<td>Elastic Solid Constitutive Examples</td>
<td>1882</td>
</tr>
<tr>
<td>404.2.1</td>
<td>Linear Elastic Constitutive Examples</td>
<td>1882</td>
</tr>
<tr>
<td>404.2.1.1</td>
<td>Pure Shear, Monotonic Loading</td>
<td>1882</td>
</tr>
<tr>
<td>404.2.1.2</td>
<td>Pure Shear, Cyclic Loading</td>
<td>1883</td>
</tr>
<tr>
<td>404.2.1.3</td>
<td>Uniaxial Strain, Monotonic Loading</td>
<td>1884</td>
</tr>
<tr>
<td>404.2.1.4</td>
<td>Uniaxial Strain, Cyclic Loading</td>
<td>1885</td>
</tr>
<tr>
<td>404.2.2</td>
<td>Nonlinear Elastic Constitutive Examples</td>
<td>1886</td>
</tr>
<tr>
<td>404.2.2.1</td>
<td>Triaxial Uniform Pressure, Monotonic Loading</td>
<td>1886</td>
</tr>
<tr>
<td>404.2.3</td>
<td>Nonlinear Elastic Constitutive Examples</td>
<td>1887</td>
</tr>
<tr>
<td>404.2.3.1</td>
<td>Triaxial Uniform Pressure, Cyclic Loading</td>
<td>1887</td>
</tr>
<tr>
<td>404.3</td>
<td>Elastic Plastic Solid Constitutive Examples</td>
<td>1888</td>
</tr>
<tr>
<td>404.3.1</td>
<td>Elastic Perfectly Plastic Constitutive Examples</td>
<td>1888</td>
</tr>
<tr>
<td>404.3.1.1</td>
<td>Pure Shear</td>
<td>1888</td>
</tr>
<tr>
<td>404.3.1.2</td>
<td>Uniaxial Strain</td>
<td>1889</td>
</tr>
<tr>
<td>404.3.2</td>
<td>Elastic Plastic, Isotropic Hardening, Constitutive Examples</td>
<td>1890</td>
</tr>
<tr>
<td>404.3.2.1</td>
<td>Pure Shear, Monotonic Loading</td>
<td>1890</td>
</tr>
<tr>
<td>404.3.2.2</td>
<td>Pure Shear, Cyclic Loading</td>
<td>1891</td>
</tr>
<tr>
<td>404.3.2.3</td>
<td>Uniaxial Strain, Monotonic Loading</td>
<td>1892</td>
</tr>
<tr>
<td>404.3.2.4</td>
<td>Uniaxial Strain, Cyclic Loading</td>
<td>1893</td>
</tr>
<tr>
<td>404.3.3</td>
<td>Elastic Plastic, Kinematic Hardening, Constitutive Examples</td>
<td>1894</td>
</tr>
<tr>
<td>404.3.3.1</td>
<td>Pure Shear, Monotonic Loading</td>
<td>1894</td>
</tr>
<tr>
<td>404.3.3.2</td>
<td>Pure Shear, Cyclic Loading</td>
<td>1895</td>
</tr>
<tr>
<td>404.3.3.3</td>
<td>Uniaxial Strain, Monotonic Loading</td>
<td>1896</td>
</tr>
<tr>
<td>404.3.3.4</td>
<td>Uniaxial Strain, Cyclic Loading</td>
<td>1897</td>
</tr>
<tr>
<td>404.3.4</td>
<td>Elastic Plastic, Armstrong-Frederick, von-Mises, Constitutive Examples</td>
<td>1898</td>
</tr>
<tr>
<td>404.3.4.1</td>
<td>Pure Shear, Cyclic Loading</td>
<td>1898</td>
</tr>
<tr>
<td>404.3.5</td>
<td>Elastic Plastic, Armstrong-Frederick, Drucker-Prager, Constitutive Examples</td>
<td>1899</td>
</tr>
<tr>
<td>404.3.5.1</td>
<td>Pure Shear, Cyclic Loading</td>
<td>1899</td>
</tr>
<tr>
<td>404.3.6</td>
<td>Elastic Plastic, SanSAND, Constitutive Examples</td>
<td>1900</td>
</tr>
<tr>
<td>404.3.6.1</td>
<td>Bardet Constraint Examples</td>
<td>1900</td>
</tr>
</tbody>
</table>
404.4 Stiffness Reduction and Damping Curves Modeling .......................... 1901

404.4.1 Multi-yield-surface von-Mises ................................................. 1901

404.4.1.1 Model description .............................................................. 1901

404.4.1.2 Real-ESSI input file ......................................................... 1901

404.4.2 Multi-yield-surface Drucker-Prager .......................................... 1904

404.4.2.1 Problem description ......................................................... 1904

404.4.2.2 Real-ESSI input file: ......................................................... 1904

404.4.3 Simulate Stiffness Reduction using von-Mises Armstrong-Frederick ......... 1907

404.4.3.1 Model description .............................................................. 1907

404.4.3.2 Real-ESSI input file: ......................................................... 1907

404.5 Cosserat, Micropolar Material Modeling ....................................... 1909

404.5.1 Cosserat, Micropolar Elastic Material Model (example in development) ...... 1910

404.5.2 Cosserat, Micropolar Elastic-Plastic von Mises Material Model (example in development) 1911

404.5.3 Cosserat, Micropolar Elastic-Plastic Druekcr Prager Material Model (example in development) 1912

405 Static Examples

(2016-2017-2019-2021-) ................................................................. 1913

405.1 Chapter Summary and Highlights ................................................. 1914

405.2 Static Elastic Solid Examples ..................................................... 1914

405.2.1 Statics, Bricks, with Nodal Forces ........................................... 1914

405.2.1.1 Statics, 8 Node Brick, with Nodal Forces ................................ 1914

405.2.1.2 Statics, 27 Node Brick, with Nodal Forces ................................ 1915

405.2.1.3 Statics, 8-27 Node Brick, with Nodal Forces ............................ 1915

405.2.2 Statics, Bricks, with Surface Loads .......................................... 1916

405.2.2.1 Statics, 8 Node Brick, with Surface Forces ................................ 1916

405.2.2.2 Statics, 27 Node Brick, with Surface Forces ............................ 1917

405.2.3 Statics, Bricks, with Body Forces ............................................ 1918

405.2.3.1 Statics, 8 Node Brick, with Body Forces ................................ 1918

405.2.3.2 Statics, 27 Node Brick, with Body Forces ................................ 1919

405.2.3.3 Statics, 8-27 Node Brick, with Body Forces ............................ 1920

405.3 Static Elastic Structural Examples .............................................. 1921

405.3.1 Statics, Truss, with Nodal Forces ............................................ 1921

405.3.2 Statics, Elastic Beam, with Nodal Forces ................................... 1922
405.3.3 Statics, Elastic Beam, with Body Forces ............................................ 1923
405.3.4 Statics, ShearBeam Element .............................................................. 1924
405.3.4.1 Problem description ................................................................. 1924
405.3.4.2 Results ................................................................................. 1924
405.3.5 Statics, Elastic Shell, with Nodal Forces ........................................... 1926
405.3.5.1 ANDES Shell, out of Plane Force ............................................ 1926
405.3.5.2 Perpendicular to Plane (bending) ............................................... 1926
405.3.5.3 ANDES Shell, In-plane Force .................................................. 1926
405.3.6 Statics, Elastic Shell, with Body Forces ........................................... 1928
405.3.6.1 ANDES shell under the out-of-Plane Body Force ..................... 1928
405.3.6.2 ANDES Shell, In-plane Body Force .......................................... 1928
405.4 Statics, Interface/Contact Elements ..................................................... 1930
405.4.1 Statics, Two Bar Normal Interface/Contact Problem Under Monotonic Loading .................................................. 1930
405.4.2 Statics, Four Bar Interface/Contact Problem With Normal and Shear Force Under Monotonic Loading .................................................. 1931
405.4.3 Statics, 3-D Truss example with normal confinement and Shear Loading .................................................. 1933
405.4.4 Statics, Six Solid Blocks Example With Interface/Contact ................. 1935
405.5 Static Inelastic Solid Examples ............................................................. 1936
405.5.1 Statics, Bricks, Elastic-Plastic, von Mises, with Nodal Forces .......... 1936
405.5.2 Statics, Bricks, Elastic-Plastic, Drucker Prager, with Nodal Forces .... 1936
405.6 Static Inelastic Shell Examples (example in development) ................... 1939
405.7 Statics, Elastic Single Solid Finite Finite Element Examples .................. 1939
405.7.1 Statics, Linear Elastic, Solid Examples ............................................. 1939
405.7.1.1 Statics, Pure Shear, Monotonic Loading .................................... 1939
405.7.1.2 Pure Shear, Cyclic Loading ...................................................... 1940
405.7.1.3 Uniaxial Strain, Monotonic Loading ......................................... 1940
405.7.1.4 Uniaxial Strain, Cyclic Loading ................................................. 1941
405.7.2 Statics, Nonlinear Elastic, Duncan-Chang, Pure Shear, Solid Examples .................. 1941
405.7.2.1 Pure Shear, Monotonic Loading .............................................. 1941
405.7.2.2 Pure Shear, Cyclic Loading ..................................................... 1942
405.8 Statics, Elastic-Plastic Single Solid Finite Element Examples ................. 1944
405.8.1 Statics, Elastic Perfectly Plastic, Cyclic Loading, Pure Shear Solid Examples .................. 1944
405.8.1.1 Statics, von Mises Yield Function, Isotropic Hardening ............... 1945
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>405.8.1.2</td>
<td>Statics, von Mises Yield Function, Kinematic Hardening</td>
<td>1946</td>
</tr>
<tr>
<td>405.8.1.3</td>
<td>Statics, Drucker Prager Yield Function, von-Mises Plastic Potential Function, Perfectly Plastic Hardening Rule</td>
<td>1947</td>
</tr>
<tr>
<td>405.8.1.4</td>
<td>Statics, Drucker Prager Yield Function, Drucker Prager Plastic Potential Function, Perfectly Plastic Hardening Rule.</td>
<td>1948</td>
</tr>
<tr>
<td>405.8.2</td>
<td>Statics, Drucker Prager with Armstrong Frederick Nonlinear Kinematic Hardening Material Model</td>
<td>1949</td>
</tr>
<tr>
<td>405.9</td>
<td>Statics, Elastic, Fiber Cross Section Beam Finite Element Examples</td>
<td>1951</td>
</tr>
<tr>
<td>405.9.1</td>
<td>Statics, Linear Elastic, Normal Loading and Pure Bending Fiber Cross Section Beam Finite Element Examples</td>
<td>1951</td>
</tr>
<tr>
<td>405.9.1.1</td>
<td>Linear Elastic Normal Loading, (Fiber Cross Section) Beam Finite Element Examples</td>
<td>1951</td>
</tr>
<tr>
<td>405.9.1.2</td>
<td>Linear Elastic Pure Bending, (Fiber Cross Section) Beam Finite Element Examples</td>
<td>1952</td>
</tr>
<tr>
<td>405.10</td>
<td>Statics, Elastic-Plastic, Fiber Cross Section Beam Finite Element Examples</td>
<td>1953</td>
</tr>
<tr>
<td>405.10.1</td>
<td>Statics, Elastic-Plastic, Normal Loading and Pure Bending Fiber Cross Section Beam Finite Element</td>
<td>1953</td>
</tr>
<tr>
<td>405.10.1.1</td>
<td>Elastic-Plastic Normal Loading, (Fiber Cross Section) Beam Finite Element Examples</td>
<td>1953</td>
</tr>
<tr>
<td>405.10.1.2</td>
<td>Elastic-Plastic Pure Bending, (Fiber Cross Section) Beam Finite Element Examples</td>
<td>1954</td>
</tr>
<tr>
<td>405.11</td>
<td>Statics, Elastic, Inelastic Wall Finite Element Examples</td>
<td>1955</td>
</tr>
<tr>
<td>405.11.1</td>
<td>Statics, Linear Elastic, Wall Finite Element Examples</td>
<td>1955</td>
</tr>
<tr>
<td>405.11.1.1</td>
<td>Statics, Linear Elastic, Wall Finite Element Examples</td>
<td>1955</td>
</tr>
<tr>
<td>405.11.1.2</td>
<td>Linear Elastic, Bi-Axial, Wall Finite Element Examples</td>
<td>1956</td>
</tr>
<tr>
<td>405.11.1.3</td>
<td>Linear Elastic, Shear, (Fiber Cross Section) Wall Finite Element Examples</td>
<td>1957</td>
</tr>
<tr>
<td>405.12</td>
<td>Statics, Elastic-Plastic Wall Finite Element Examples</td>
<td>1958</td>
</tr>
<tr>
<td>405.12.1</td>
<td>Statics, Elastic-Plastic, in Plane, Wall Finite Element Examples</td>
<td>1958</td>
</tr>
<tr>
<td>405.12.1.2</td>
<td>Elastic-Plastic, Bi-Axial, Wall Finite Element Examples</td>
<td>1959</td>
</tr>
<tr>
<td>405.12.1.3</td>
<td>Elastic-Plastic, Shear, Wall Finite Element Examples</td>
<td>1960</td>
</tr>
<tr>
<td>405.13</td>
<td>Statics, Solution Advancement Control</td>
<td>1961</td>
</tr>
<tr>
<td>405.13.1</td>
<td>Increments: Load Control</td>
<td>1961</td>
</tr>
<tr>
<td>405.13.1.1</td>
<td>Solids Example, Elastic Plastic Isotropic Hardening</td>
<td>1961</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>405.13.1.2 Solids Example, Elastic Plastic Kinematic Hardening</td>
<td>1962</td>
<td></td>
</tr>
<tr>
<td>405.13.1.3 Inelastic Beam Example, Steel and Reinforced Concrete</td>
<td>1963</td>
<td></td>
</tr>
<tr>
<td>405.13.2 Statics, Increments: Displacement Control</td>
<td>1963</td>
<td></td>
</tr>
<tr>
<td>405.13.2.1 Statics, Single Displacement Control</td>
<td>1963</td>
<td></td>
</tr>
<tr>
<td>405.13.2.2 Solids Example, Elastic-Perfectly Plastic</td>
<td>1963</td>
<td></td>
</tr>
<tr>
<td>405.13.2.3 Solids Example, Elastic Plastic Isotropic Hardening</td>
<td>1964</td>
<td></td>
</tr>
<tr>
<td>405.13.2.4 Solids Example, Elastic Plastic Kinematic Hardening</td>
<td>1964</td>
<td></td>
</tr>
<tr>
<td>405.13.2.5 Inelastic Beam Example, Steel and Reinforced Concrete</td>
<td>1966</td>
<td></td>
</tr>
<tr>
<td>405.13.3 Statics, Solution Algorithms</td>
<td>1968</td>
<td></td>
</tr>
<tr>
<td>405.13.3.1 Statics, Solution Algorithm: No Convergence Check</td>
<td>1968</td>
<td></td>
</tr>
<tr>
<td>405.13.3.2 Statics, Solution Algorithm: Newton Algorithm</td>
<td>1969</td>
<td></td>
</tr>
<tr>
<td>405.13.3.3 Statics, Solution Algorithm: Newton Algorithm with Line Search</td>
<td>1970</td>
<td></td>
</tr>
<tr>
<td>405.13.4 Statics, Solution Advancement Control, Iterations: Convergence Criteria</td>
<td>1971</td>
<td></td>
</tr>
<tr>
<td>405.13.4.1 Statics, Solution Advancement Control, Convergence Criteria: Unbalanced Force</td>
<td>1971</td>
<td></td>
</tr>
<tr>
<td>405.13.4.2 Statics, Solution Advancement Control, Convergence Criteria: Displacement Increment</td>
<td>1972</td>
<td></td>
</tr>
<tr>
<td>405.13.5 Statics, Solution Advancement Control, Different Convergence Tolerances</td>
<td>1973</td>
<td></td>
</tr>
<tr>
<td>(Examples in preparation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>405.14 Statics, Small Practical Examples</td>
<td>1974</td>
<td></td>
</tr>
<tr>
<td>405.14.1 Statics, Elastic Beam Element for a Simple Frame Structure</td>
<td>1974</td>
<td></td>
</tr>
<tr>
<td>405.14.1.1 Problem Description</td>
<td>1974</td>
<td></td>
</tr>
<tr>
<td>405.14.2 Statics, 4NodeANDES Square Plate, Four Edges Clamped</td>
<td>1975</td>
<td></td>
</tr>
<tr>
<td>405.14.2.1 Problem description:</td>
<td>1975</td>
<td></td>
</tr>
<tr>
<td>405.14.2.2 Numerical model:</td>
<td>1975</td>
<td></td>
</tr>
<tr>
<td>405.14.3 Statics, Six Solid Blocks Example With Interface/Contact</td>
<td>1977</td>
<td></td>
</tr>
</tbody>
</table>

406 Dynamic Examples


<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>406.1 Chapter Summary and Highlights</td>
<td>1980</td>
</tr>
<tr>
<td>406.2 Dynamic Solution Advancement (in Time)</td>
<td>1980</td>
</tr>
<tr>
<td>406.2.1 Dynamics: Newmark Method</td>
<td>1980</td>
</tr>
<tr>
<td>406.2.1.1 Model Description</td>
<td>1980</td>
</tr>
<tr>
<td>406.2.1.2 Results</td>
<td>1980</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>406.2.2 Dynamics: Hilber-Hughes-Taylor ($\alpha$) Method.</td>
<td>1981</td>
</tr>
<tr>
<td>406.2.2.1 Model Description</td>
<td>1981</td>
</tr>
<tr>
<td>406.2.2.2 Results</td>
<td>1981</td>
</tr>
<tr>
<td>406.3 Dynamics: Solution Advancement: Time Step Size</td>
<td>1982</td>
</tr>
<tr>
<td>406.3.1 Dynamics: Solution Advancement: Equal Time Step</td>
<td>1982</td>
</tr>
<tr>
<td>406.3.1.1 Model Description</td>
<td>1982</td>
</tr>
<tr>
<td>406.3.1.2 Results</td>
<td>1982</td>
</tr>
<tr>
<td>406.3.2 Dynamics Solution Advancement: Variable Time Step</td>
<td>1983</td>
</tr>
<tr>
<td>406.3.2.1 Model Description</td>
<td>1983</td>
</tr>
<tr>
<td>406.3.2.2 Results</td>
<td>1983</td>
</tr>
<tr>
<td>406.4 Dynamics: Energy Dissipation, Damping</td>
<td>1984</td>
</tr>
<tr>
<td>406.4.1 Dynamics: Energy Dissipation: Viscous Damping</td>
<td>1984</td>
</tr>
<tr>
<td>406.4.1.1 Dynamics: Energy Dissipation, Viscous Damping: Rayleigh Damping</td>
<td>1984</td>
</tr>
<tr>
<td>406.4.1.2 Dynamics: Energy Dissipation, Viscous Damping: Caughey Damping</td>
<td>1984</td>
</tr>
<tr>
<td>406.4.2 Dynamics: Energy Dissipation: Material (Elastic-Plastic, Hysteretic) Damping</td>
<td>1986</td>
</tr>
<tr>
<td>406.4.2.1 Dynamics: Energy Dissipation, Material Damping: Elastic Perfectly Plastic Models</td>
<td>1986</td>
</tr>
<tr>
<td>406.4.2.2 Dynamics: Energy Dissipation, Material/Hysteretic Damping: Elastic Plastic Isotropic Hardening Models</td>
<td>1987</td>
</tr>
<tr>
<td>406.4.2.3 Dynamics: Energy Dissipation, Material/Hysteretic Damping: Elastic Plastic Kinematic Hardening Models</td>
<td>1987</td>
</tr>
<tr>
<td>406.4.2.4 Dynamics: Energy Dissipation, Material/Hysteretic Damping: Elastic Plastic Armstrong-Frederick Models</td>
<td>1988</td>
</tr>
<tr>
<td>406.4.3 Dynamics: Energy Dissipation: Numerical Damping</td>
<td>1989</td>
</tr>
<tr>
<td>406.4.3.1 Energy Dissipation, Numerical Damping: Newmark Method</td>
<td>1989</td>
</tr>
<tr>
<td>406.4.3.2 Dynamics: Energy Dissipation, Numerical Damping: Hilber-Hughes-Taylor ($\alpha$) Method</td>
<td>1990</td>
</tr>
<tr>
<td>406.5 Dynamics: Elastic Solid Dynamic Examples.</td>
<td>1991</td>
</tr>
<tr>
<td>406.5.1 Model Description</td>
<td>1991</td>
</tr>
<tr>
<td>406.5.2 Results</td>
<td>1991</td>
</tr>
<tr>
<td>406.6 Dynamics: Elastic Structural Dynamic Examples.</td>
<td>1992</td>
</tr>
<tr>
<td>406.6.1 Model Description</td>
<td>1992</td>
</tr>
<tr>
<td>406.6.2 Results</td>
<td>1992</td>
</tr>
</tbody>
</table>
406.7 Dynamics: Interface/Contact Elements ................. 1994
406.7.1 Dynamics: Hard Interface/Contact: One Bar Normal Interface/Contact Dynamics ................. 1994
406.7.1.1 Model Description .................................. 1994
406.7.1.2 Dynamics: No Viscous Damping ................. 1995
406.7.1.3 Dynamics: Normal Viscous Damping Between Interface/Contact Node Pairs ................. 1995
406.7.1.4 Dynamics: Explicit Simulation ................. 1996
406.7.2 Dynamics: Hard Interface/Contact: Frictional Single Degree of Freedom Problem ................. 1997
406.7.2.1 Dynamics: No Viscous Damping ................. 1998
406.7.2.2 Dynamics: Tangential Viscous Damping Between Interface/Contact Node Pairs ................. 1998
406.7.2.3 Dynamics: Explicit Simulation ................. 1999
406.7.3 Dynamics: Soft Interface/Contact: One Bar Normal Interface/Contact Dynamics 2000
406.7.3.1 Dynamics: No Viscous Damping ................. 2001
406.7.3.2 Dynamics: With Normal Viscous Damping Between Interface/Contact Node Pairs ................. 2001
406.7.3.3 Dynamics: Explicit Simulation ................. 2002
406.7.4 Dynamics: Soft Interface/Contact: Frictional Single Degree of Freedom Problem ................. 2003
406.7.4.1 Dynamics: No Viscous Damping ................. 2004
406.7.4.2 Dynamics: Tangential Viscous Damping Between Interface/Contact Node Pairs ................. 2004
406.7.4.3 Dynamics: Explicit Simulation ................. 2005
406.7.5 Dynamics: Split Beam ............................. 2006
406.7.5.1 Model Description ......................... 2006
406.7.5.2 Dynamics: Split Beam With Hard Interface/Contact ......... 2006
406.7.5.3 Dynamics: Split Beam With Soft Interface/Contact ................. 2007
406.7.6 Dynamics: Block on Soil ESSI ................. 2008
406.8 Dynamics: Inelastic Solid Examples ................. 2010
406.9 Dynamics: Inelastic Structural Examples ................. 2011
406.10 Dynamics: Domain Reduction Method (DRM) .................................. 2012
406.10.1 Dynamics: DRM One Dimensional (1D) Model ......................... 2012
406.10.2 Dynamics: Three Dimensional (3D) DRM Model ...................... 2016
406.10.3 Dynamics: DRM Model with Structure .................................. 2018
406.11 Dynamics: Eigen Analysis .......................................................... 2019
406.12 Dynamics: Fully Coupled u-p-U Element (example in development) . 2020
406.13 Dynamics: Partially Saturated / Unsaturated u-p-U Element (example in development) . 2021
406.14 Dynamics: Fully Coupled u-p Element (example in development) . 2022
406.15 Dynamics: Coupled Interface/Contact Element (example in development) . 2023
406.16 Dynamics: Buoyant Forces (example in development) .................. 2024

407 Stochastic Examples

(2018-2019-2020-2021-) ........................................................................ 2025
407.1 Chapter Summary and Highlights ................................................. 2026
407.2 Probabilistic Constitutive Modeling .............................................. 2026
407.2.1 Probabilistic Constitutive Modeling: Linear Elastic .................. 2026
407.2.2 Probabilistic Constitutive Modeling: Elasto-Plastic .................. 2027
407.3 Probabilistic Characterization of Seismic Motions ......................... 2027
407.4 1D Stochastic Seismic Wave Propagation .................................... 2028
407.4.1 1D Stochastic Seismic Wave Propagation: Linear Elastic .......... 2028
407.4.2 1D Stochastic Seismic Wave Propagation: Elasto-Plastic .......... 2031
407.5 1D Stochastic Seismic Wave Propagation: Sobol Sensitivity Analysis . 2031

408 Three Day Short Course Examples

(2017-2021-) ....................................................................................... 2033
408.1 Nonlinear Analysis Steps ......................................................... 2034
408.1.1 Free Field 1C ....................................................................... 2034
408.1.2 Free Field 3C ....................................................................... 2038
408.1.3 Soil-Foundation Interaction 3D .............................................. 2042
408.1.4 Soil-Structure Interaction 3D ................................................. 2046
408.1.5 Analysis of a Structure without Soil ...................................... 2053
408.1.5.1 Eigen Analysis ................................................................. 2053
408.1.5.2 Imposed Motion ............................................................. 2055
408.2 Day 1: Overview ........................................................................ 2058
408.2.1 Nuclear Power Plant with 3C motions from SW4 ................... 2058
408.2.2 Nuclear Power Plant with 1C motions from Deconvolution .......... 2060
408.2.3 Nuclear Power Plant with $3 \times 1C$ motions from Deconvolution. 2062
408.2.4 Single Element Models: Illustration of the Elastic-Plastic Behavior 2064
408.2.5 Pushover for Nonlinear Frame. 2066
408.2.6 Pre-Processing examples with Gmsh 2068
  408.2.6.1 Cantilever Example 2068
  408.2.6.2 Brick-shell-beam Example 2069
  408.2.6.3 DRM 2D Example 2071
  408.2.6.4 DRM 3D Example 2072
408.2.7 Post-processing examples with ParaView 2073
  408.2.7.1 Slice Visualization 2073
  408.2.7.2 Stress Visualization 2074
  408.2.7.3 Pore Pressure Visualization with upU Element 2075
  408.2.7.4 Eigen Visualization. 2076
408.2.8 Check Model and Visualization of Boundary Conditions 2077
408.2.9 Restart Simulation 2078
  408.2.9.1 Restart in the next stage 2078
  408.2.9.2 Restart inside the stage 2079
408.3 Day 2: Seismic Motions 2080
  408.3.1 Deconvolution and Propagation of 1C Motions, 1D Model 2080
  408.3.2 Convolution and Propagation of 1C Motions, 1D Model 2080
  408.3.3 Convolution, Deconvolution and Propagation of 1C Motions, 2D Model 2080
  408.3.3.1 ESSI 3D building model, deconvolution 1C model, shell model with DRM 2082
  408.3.4 Deconvolution $3 \times 1C$ Motions. 2084
  408.3.4.1 Free field 1C model, deconvolution $3 \times 1C$ motion, model with DRM 2084
  408.3.4.2 Free field 3D model, deconvolution $3 \times 1C$ motion, model with DRM 2086
  408.3.4.3 ESSI 3D building model, deconvolution $3 \times 1C$ motion, shell model with DRM 2087
  408.3.5 Mesh Dependence of Wave Propagation Frequencies 2089
  408.3.6 Application of 3C Motions from SW4 2091
  408.3.6.1 3C Seismic Motion from SW4 2091
  408.3.6.2 Free field 3D model, 3C motion, model with DRM 2093
  408.3.6.3 ESSI 3D building model, 3C motion, shell model with DRM 2095
408.4 Day 3: Inelastic, Nonlinear Analysis

408.4.1 Single Element Models: Illustration of the Elastic-Plastic Behavior

408.4.1.1 von-Mises Perfectly Plastic Material Model

408.4.1.2 von-Mises Armstrong-Frederick Material Model

408.4.1.3 von-Mises G/Gmax Material Model

408.4.1.4 Drucker-Prager Perfectly Plastic Material Model

408.4.1.5 Drucker-Prager Armstrong-Frederick Non-Associated Material Model

408.4.1.6 Drucker-Prager G/Gmax Non-Associated Material Model

408.4.2 Wave Propagation Through Elasto-plastic Soil

408.4.3 Contact/Interface/Joint Examples

408.4.3.1 Axial Behavior: Stress-Based Hard Contact/Interface/Joint Example

408.4.3.2 Axial Behavior: Stress-Based Soft Contact/Interface/Joint Example

408.4.3.3 Shear behavior: Stress-Based Elastic Perfectly Plastic Contact/Interface/Joint

408.4.3.4 Shear behavior: Stress-Based Elastic-Hardening Contact/Interface/Joint

408.4.3.5 Shear behavior: Stress-Based Elastic-Hardening-Softening Contact/Interface/Joint

408.4.3.6 Force Based Contact/Interface/Joint Example: Base Isolator

408.4.4 Inelastic Frame Pushover

408.4.5 Inelastic Wall Pushover

408.4.6 Viscous Nonlinear behavior

408.4.7 Numerical Damping Example

408.4.8 Nuclear Power Plant Example with Nonlinearities

408.4.9 Buildings, ATC-144/FEMA-P-2091 Examples

409 Online Short Course Examples

(2019-2020-2021-)

409.1 Real-ESSI Simulator Setup

409.2 Phased Modeling of ESSI: Nonlinear Analysis Steps

409.3 Week #1: Nonlinear/Inelastic Modeling Overview

409.4 Week #2: Discussion, Nonlinear/Inelastic Modeling

409.5 Week #3: Seismic Motions

409.6 Week #4: Discussion on Seismic Motions

409.7 Week #5: Inelastic Material and Contacts/Interfaces

409.8 Week #6: Discussion, Inelastic Material and Contacts/Interfaces
500  Application to Practical Engineering Problems

501  Static Soil-Pile and Soil-Pile Group Interaction in Single Phase Soils

501.1  Chapter Summary and Highlights


501.2.1  Introduction

501.2.2  Constitutive Models

501.2.3  Simulation Results

501.2.3.1  Pile Models

501.2.3.2  Plastic Zones

501.2.3.3  \( p - y \) Curves

501.2.3.4  Comparisons of Pile Behavior in Uniform and Layered Soils

501.2.3.5  Comparison to Centrifuge Tests and LPile Results

501.2.4  Summary

501.3  Study of soil layering effects on lateral loading behavior of piles

501.3.1  Introduction

501.3.2  Finite Element Pile Models.

501.3.3  Constitutive Models

501.3.4  Comparison of \( p - y \) Behavior in Uniform and Layered Soil Deposits

501.3.5  Parametric Study for the Lateral Resistance Ratios in Terms of Stiffness and Strength Parameters.

501.3.6  Summary

501.4  Numerical Study of Group Effects for Pile Groups in Sands

501.4.1  Introduction

501.4.2  Pile Models

501.4.3  Summary of Centrifuge Tests

501.4.4  Finite Element Pile Models.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>501.4.5</td>
<td>Simulation Results</td>
<td>2177</td>
</tr>
<tr>
<td>501.4.6</td>
<td>Plastic Zone</td>
<td>2178</td>
</tr>
<tr>
<td>501.4.7</td>
<td>Bending Moment</td>
<td>2180</td>
</tr>
<tr>
<td>501.4.8</td>
<td>Load Distribution</td>
<td>2185</td>
</tr>
<tr>
<td>501.4.9</td>
<td>$p - y$ Curve</td>
<td>2190</td>
</tr>
<tr>
<td>501.4.10</td>
<td>Comparison with the Centrifuge Tests</td>
<td>2190</td>
</tr>
<tr>
<td>501.4.11</td>
<td>Conclusions</td>
<td>2197</td>
</tr>
</tbody>
</table>

### Earthquake Soil Structure Interaction, General Aspects


<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>502.1</td>
<td>Chapter Summary and Highlights</td>
<td>2198</td>
</tr>
<tr>
<td>502.2</td>
<td>Free Field Ground Motions</td>
<td>2199</td>
</tr>
<tr>
<td>502.2.1</td>
<td>Seismic Motions: Available Data</td>
<td>2201</td>
</tr>
<tr>
<td>502.2.2</td>
<td>Multi-Directional and Seismic Input Coming in at Inclined Angle</td>
<td>2204</td>
</tr>
<tr>
<td>502.2.3</td>
<td>Free Field Motion Development</td>
<td>2205</td>
</tr>
<tr>
<td>502.2.3.1</td>
<td>Details of Free Field Motion Development</td>
<td>2205</td>
</tr>
<tr>
<td>502.2.4</td>
<td>Free Field Ground Motions Development: Closed Form Solution</td>
<td>2209</td>
</tr>
<tr>
<td>502.2.4.1</td>
<td>Three Component, 3C Motion Development</td>
<td>2209</td>
</tr>
<tr>
<td>502.2.4.2</td>
<td>One Component, 1C Motion Development</td>
<td>2209</td>
</tr>
<tr>
<td>502.2.5</td>
<td>Free Field Ground Motions Development: Frequency Wave Number Method</td>
<td>2211</td>
</tr>
<tr>
<td>502.2.5.1</td>
<td>(Green’s functions) (fk)</td>
<td>2211</td>
</tr>
<tr>
<td>502.2.6</td>
<td>Free Field Ground Motions Development: Fault Slip Model</td>
<td>2217</td>
</tr>
<tr>
<td>502.2.6.1</td>
<td>Input Motion</td>
<td>2223</td>
</tr>
<tr>
<td>502.2.6.2</td>
<td>Select Seismic Motions, Displacement Array Traces</td>
<td>2225</td>
</tr>
<tr>
<td>502.2.6.3</td>
<td>Animations of Fault Slip Motions</td>
<td>2269</td>
</tr>
<tr>
<td>502.2.6.4</td>
<td>Point Fault Slip Motions, Arrays and Particle Motions</td>
<td>2270</td>
</tr>
<tr>
<td>502.3</td>
<td>Dynamic Soil-Foundation-Structure Interaction</td>
<td>2271</td>
</tr>
<tr>
<td>502.3.0.1</td>
<td>Animation of the DRM on a 1D Stack of Elements</td>
<td>2271</td>
</tr>
<tr>
<td>502.3.0.2</td>
<td>Using External Finite-Difference Seismic Code for DRM Motions</td>
<td>2272</td>
</tr>
<tr>
<td>502.3.1</td>
<td>Seismic Wave Propagation Modeling and Simulation: Numerical Accuracy</td>
<td>2283</td>
</tr>
<tr>
<td>502.3.1.1</td>
<td>Grid Spacing $\Delta h$</td>
<td>2283</td>
</tr>
<tr>
<td>502.3.1.2</td>
<td>Time Step Length $\Delta t$</td>
<td>2284</td>
</tr>
<tr>
<td>502.3.1.3</td>
<td>Nonlinear Material Models</td>
<td>2284</td>
</tr>
</tbody>
</table>
502.3.2 Seismic Wave Propagation Modeling and Simulation: Domain Boundaries  2285
502.3.3 Soil/Rock Modeling and Simulation  2287
502.3.4 Soil/Rock – Foundation Contact (Slipping and Gaping) Modeling and Simulation  2287
502.3.5 Buoyancy Modeling and Simulation  2287
502.3.6 Structural Foundations Modeling and Simulation  2290
502.3.7 Seismic Isolator Modeling and Simulation  2291
502.3.8 Structural Components Modeling and Simulation  2292
502.3.9 Nonlinear Time Domain Analysis Progress and Example  2293
  502.3.9.1 Model Development  2293
  502.3.9.2 Simulation Development  2293
  502.3.9.3 Seismic Motions  2293
502.4 Step by Step, Hierarchical Inelastic ESSI Modeling and Simulation  2293

503 Earthquake-Soil-Structure Interaction, Bridge Structures

(2003-2007-2011-)  2297

503.1 Chapter Summary and Highlights  2298

503.2 Case History: Earthquake-Soil-Structure Interaction for a Bridge System  2299
  503.2.1 Prototype Bridge Model Simulation  2299
    503.2.1.1 Soil Model  2299
    503.2.1.2 Element Size Determination  2300
    503.2.1.3 Time Step Length Requirement  2304
    503.2.1.4 Domain Reduction Method  2306
    503.2.1.5 Structural Model  2306
    503.2.1.6 Simulation Scenarios  2310
  503.2.2 Earthquake Simulations - 1994 Northridge  2312
    503.2.2.1 Input Motion  2312
    503.2.2.2 Displacement Response  2314
    503.2.2.3 Acceleration Response  2318
    503.2.2.4 Displacement Response Spectra  2321
    503.2.2.5 Acceleration Response Spectra  2325
    503.2.2.6 Structural Response  2328
  503.2.3 Earthquake Simulations - 1999 Turkey Kocaeli  2332
    503.2.3.1 Displacement Response  2334
    503.2.3.2 Acceleration Response  2338
503.2.3.3 Displacement Response Spectra ........................................ 2342
503.2.3.4 Acceleration Response Spectra ....................................... 2346
503.2.3.5 Structural Response ..................................................... 2350
503.2.4 Earthquake Soil Structure Interaction Effects ....................... 2354
  503.2.4.1 How Strength of Soil Foundations Affects ESSI ................. 2354
  503.2.4.2 How Site Non-Uniformity Affects ESSI .......................... 2360
  503.2.4.3 How Input Motion Affects ESSI .................................... 2381

504 Earthquake-Soil-Structure Interaction, Nuclear Power Plants


504.1 Stick/Solid Finite Element Model ......................................... 2391
  504.1.1 Slipping behavior of SFSI models by considering 1C wave propagation ........................................ 2393
    504.1.1.1 Morgan Hill earthquake .......................................... 2394
    504.1.1.2 Ricker wave ...................................................... 2400
    504.1.2 Slipping behavior of SFSI models by considering 3C wave propagation ........................................ 2406
     504.1.2.1 Ricker wave, with fault located at 45° towards the top middle point of the model .......................... 2408
     504.1.2.2 Ricker wave, with fault located at 34° towards the top middle point of the model .......................... 2415

504.2 Three Dimensional (3C) Seismic Wave Fields and Behavior of Nuclear Power Plants (NPPs). ........................................ 2422
  504.2.1 Development of Seismic Motions: Large Scale Free Field Model ........................................ 2422
  504.2.2 NPP Response, Model #01 ............................................. 2422
  504.3 3D Representative NPP Structure Model(s) ............................. 2423
    504.3.1 Model #01, Single NPP ............................................ 2424
    504.3.2 Model #02, Single NPP ............................................ 2429
    504.3.3 Model #03, Double NPP, Soil-Structure-Soil-Structure Interaction ........................................ 2434
    504.3.4 Model #04, Small Modular Reactor (SMR) ......................... 2441

504.4 3C (6C) vs 1C Seismic Motions ............................................ 2442
  504.4.1 Appropriate Use of 3C and 3×1C and 1C Seismic Motions ........ 2442
  504.4.2 Illustration of Use of 3C and 1C Seismic Motions ................ 2443

504.5 3C (6C) vs 3×1C vs 1C Seismic Motions ................................ 2451

504.6 3D Nonlinear Modeling for Nuclear Power Plants ....................... 2452
  504.6.1 Introduction ............................................................. 2452
504.6.2 Model Development and Simulation Details ........................................... 2453
  504.6.2.1 Structure Model. ................................................................. 2454
  504.6.2.2 Soil Model. ................................................................. 2454
  504.6.2.3 Interface/Contact Modeling ............................................. 2455
  504.6.2.4 Seismic Motions. ......................................................... 2456
  504.6.2.5 Domain Reduction Method ............................................. 2457
  504.6.2.6 Staged Simulation ......................................................... 2458
  504.6.3 Simulation Results .............................................................. 2458
  504.6.4 Energy Dissipation .............................................................. 2462
  504.6.5 Conclusion ................................................................. 2465

504.7 3D Nonlinear Modeling for Small Modular Reactors (SMRs) ................. 2466
  504.7.1 Introduction ................................................................. 2467
  504.7.2 Domain Reduction method ................................................. 2469
  504.7.3 3C Free Field Motions ..................................................... 2470
  504.7.4 Model Description ............................................................ 2471
    504.7.4.1 Embedded Nuclear Structure ....................................... 2472
    504.7.4.2 Soil Model ........................................................... 2472
    504.7.4.3 Soft Contact Element ............................................... 2472
    504.7.4.4 Simulation Procedure ............................................... 2472
  504.7.5 Simulation Results ............................................................ 2473
  504.7.6 Energy Dissipation .............................................................. 2478
  504.7.7 Conclusion ................................................................. 2481

504.8 Three Dimensional (3D) Inelastic Modeling for Structure Soil Structure Interaction

2482

505 Liquefaction and Cyclic Mobility ............................................................... 2483
(2002-2006-2009-2021-) ............................................................................. 2483

505.1 Chapter Summary and Highlights ....................................................... 2484
505.2 Introduction ..................................................................................... 2484
505.3 Liquefaction of Level and Sloping Grounds ......................................... 2487
  505.3.1 Model Description ............................................................. 2487
  505.3.2 Behavior of Saturated Level Ground ...................................... 2488
  505.3.3 Behavior of Saturated Sloping Ground .................................... 2490
## 505.4 Pile in Liquefied Ground, Staged Simulation Model Development

- **505.4.1** First Loading Stage: Self Weight .................................................. 2494
- **505.4.2** Second Loading Stage: Pile–Column Installation .......................... 2496
- **505.4.3** Third Loading Stage: Seismic Shaking ........................................... 2498
- **505.4.4** Free Field, Lateral and Longitudinal Models ................................. 2498

## 505.5 Simulation Results

- **505.5.1** Pore Fluid Migration ................................................................. 2499
- **505.5.2** Soil Skeleton Deformation ........................................................... 2503
- **505.5.3** Pile Response ................................................................................. 2506
- **505.5.4** Pile Pinning Effects ....................................................................... 2507

## 506 Slope Stability in 2D and 3D

(1999-2010-)

- **506.1** Chapter Summary and Highlights .................................................... 2511
- **506.2** Introduction ..................................................................................... 2511
- **506.3** Dam Section Geometry ................................................................... 2511
- **506.4** Finite Element Modeling ................................................................. 2517
  - **506.4.1** Material Models ........................................................................ 2517
  - **506.4.2** Two Dimensional Models .......................................................... 2518
  - **506.4.3** Three Dimensional Models ....................................................... 2520
  - **506.4.4** Modeling Issues ....................................................................... 2524
- **506.5** Results: Factors of Safety ............................................................... 2526
- **506.6** Uncertainty of Results ..................................................................... 2527
- **506.7** Conclusion ..................................................................................... 2527
- **506.8** Displacement Patterns ..................................................................... 2528

## 507 Concrete Structures


- **507.1** Chapter Summary and Highlights .................................................... 2532
- **507.2** Concrete Wall/Membrane ................................................................. 2532
  - **507.2.1** Introduction ............................................................................. 2532
  - **507.2.1.1** Motivation ........................................................................... 2532
  - **507.2.2** Model Development ................................................................. 2533
  - **507.2.2.1** Model Mesh ........................................................................ 2533
  - **507.2.2.2** Plastic Damage Concrete Material Model ............................ 2536
  - **507.2.2.3** Uniaxial Steel Material Model ............................................... 2536
## 507.2.2.4 Material Model Parameters

Content...

## 507.2.3 Modeling of Energy Storage and Dissipation

### 507.2.3.1 Plastic Damage Concrete Material

Content...

### 507.2.4 Uniaxial Steel Material

Content...

### 507.2.5 Modeling and Simulation Results

#### 507.2.5.1 Force–Displacement Response

Content...

#### 507.2.5.2 Strain and Stress Distribution

Content...

#### 507.2.5.3 Concrete Damage and Energy Dissipation

Content...

## 508 ESSI for Concrete Dams

(2019-2020-2021-)

### 508.1 Chapter Summary and Highlights

Content...

### 508.2 Pine Flat Dam

Content...

## 509 ESSI for Buildings

(2018-2019-2020-2021-)

### 509.1 2D Frame with Energy Dissipation

Content...

### 509.2 Ventura Hotel

Content...

#### 509.2.1 Finite Element Model

Content...

### 509.3 Loma Linda Hospital

Content...

#### 509.3.1 Finite Element Model

Content...

### 509.4 ASCE-7 Model, Low, Steel Building

Content...

#### 509.4.1 Finite Element Model

Content...

### 509.5 ASCE-7 Model, High, Concrete Building

Content...

#### 509.5.1 Finite Element Model

Content...


### 510.1 Motivation: Modeling and Simulation of Earthquake Soil Structure Interaction

Content...

### 510.2 Introduction

Content...

### 510.3 Seismic Energy Input and Dissipation

#### 510.3.1 Seismic Energy Input

Content...

#### 510.3.2 Seismic Energy Dissipation

Content...

#### 510.3.2.1 Seismic Energy Dissipation, Wave Reflection and Wave Radiation

Content...

#### 510.3.2.2 Seismic Energy Dissipation, Viscous Coupling

Content...

#### 510.3.2.3 Seismic Energy Dissipation, Material Inelasticity

Content...
510.3.2.4 Seismic Energy Dissipation, Numerical, Algorithmic Positive and Negative Damping

510.4 Modeling: Seismic Motions

510.4.1 Seismic Motions: Available Data

510.4.2 Seismic Motion Development

510.4.2.1 Seismic Motions from Empirical Models

510.4.2.2 Seismic Motions from Geophysical Models

510.4.2.3 Seismic Motions from 3D/3C Analytic Models

510.4.2.4 Seismic Motions from Full Waveform Inversion

510.4.3 6C vs 3C vs 3×1C vs 1C Seismic Motions

510.4.4 Incoherent Seismic Motions

510.4.5 Seismic Motion Input into FEM Models

510.5 Modeling: Inelastic, Nonlinear Material Modeling for Solids and Structures

510.5.1 Inelastic Material Modeling of Rock

510.5.1.1 Calibration of Inelastic Material Model Parameters for Rock

510.5.2 Inelastic Material Modeling of Soil

510.5.2.1 Dry Soil

510.5.2.2 Fully Saturated Soil

510.5.2.3 Partially Saturated, Unsaturated Soil

510.5.2.4 Calibration of Inelastic Material Model Parameters for Soil

510.5.3 Inelastic Material Modeling of Steel

510.5.3.1 Calibration of Inelastic Material Model Parameters for Steel

510.5.4 Inelastic Material Modeling of Concrete

510.5.4.1 Calibration of Inelastic Material Model Parameters for Concrete

510.6 Modeling: Inelastic, Nonlinear Material Modeling for Contacts, Interfaces, and Joints

510.6.1 Material Modeling of Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

510.6.1.1 Calibration of Inelastic Material Model Parameters for Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

510.6.2 Material Modeling of Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

510.6.2.1 Calibration of Inelastic Material Model Parameters for Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)
510.7 Modeling: Buoyancy .................................................. 2580
510.8 Modeling: Base Isolator and Base Dissipator Systems .......... 2581
  510.8.1 Base Isolator Systems ............................................. 2581
  510.8.1.1 Calibration of Elastic/Inelastic Material Model Parameters for Base Isolator Systems ........................................ 2581
  510.8.2 Base Dissipator Systems ......................................... 2581
  510.8.2.1 Calibration of Elastic/Inelastic Material Model Parameters for Base Dissipator Systems ........................................ 2581
510.9 Modelling: Finite Element System .................................. 2582
  510.9.1 Mass Matrix ........................................................ 2582
  510.9.1.1 Consistent Mass Matrix ...................................... 2582
  510.9.1.2 Lumped Mass Matrix ........................................... 2582
  510.9.2 Viscous Damping Matrix ........................................ 2582
  510.9.2.1 Rayleigh Damping .............................................. 2582
  510.9.2.2 Caughey Damping .............................................. 2582
  510.9.3 Stiffness Matrix .................................................. 2582
  510.9.3.1 Tangent Stiffness Matrix ..................................... 2582
  510.9.3.2 Consistent Stiffness Matrix ................................... 2582
510.10 Modeling: Solid, Structure – Fluid Interaction Modeling ..... 2583
510.11 Simulation: Nonlinear Finite Elements ............................ 2584
  510.11.1 Time Marching Algorithms for Solution of Nonlinear Equations of Motion ........................................ 2584
  510.11.1.1 Newmark Algorithm ......................................... 2584
  510.11.1.2 Hilber Hughes Taylor $\alpha$ Algorithm ...................... 2584
  510.11.2 Solution of Elastic-Plastic Constitutive Equations .......... 2584
  510.11.2.1 Explicit Integration of Elastic-Plastic Constitutive Equations ........................................ 2584
  510.11.2.2 Implicit Integration of Elastic-Plastic Constitutive Equations ........................................ 2584
510.12 Modelling Guide for ESSI .......................................... 2585
  510.12.1 Buildings and NPPs on Shallow Foundations, Models ...... 2585
  510.12.2 Buildings and NPPs on Deeply Embedded Foundation (SMRs), Models ........................................ 2585
  510.12.3 Buildings and NPPs on Piles and Pile Group Foundations, Models ........................................ 2585
510.13 Practical Steps for Inelastic ESSI Analysis ........................ 2586
  510.13.1 Model Development for ESSI ................................... 2586
  510.13.2 Earthquake Soil Structure Interaction: Model Analysis ..... 2586
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>510.13.3 Earthquake Soil Structure Interaction: Results Postprocessing</td>
<td>2586</td>
</tr>
<tr>
<td>510.14 Quality Assurance Procedures for ESSI Modeling and Simulation</td>
<td>2587</td>
</tr>
<tr>
<td>510.14.1 Verification</td>
<td>2587</td>
</tr>
<tr>
<td>510.14.2 Validation</td>
<td>2587</td>
</tr>
<tr>
<td>510.15 Practical Examples, Nonlinear, Inelastic ESSI</td>
<td>2588</td>
</tr>
<tr>
<td>510.15.1 Nuclear Power Plant, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions</td>
<td>2588</td>
</tr>
<tr>
<td>510.15.2 Nuclear Power Plant on Piles, Inelastic Structure, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions</td>
<td>2588</td>
</tr>
<tr>
<td>510.15.3 Nuclear Power Plant, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 6C/3C/3×1C/1C Seismic Motions</td>
<td>2588</td>
</tr>
<tr>
<td>510.15.4 Small Modular Reactor, Deeply Embedded, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions</td>
<td>2588</td>
</tr>
<tr>
<td>510.15.5 Small Modular Reactor, Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil (Cyclic Mobility and Liquefaction), Inelastic Saturated Contact/Interface (Buoyant Pressures), 6C/3C/3×1C/1C Seismic Motions</td>
<td>2588</td>
</tr>
<tr>
<td>510.15.6 Multiple Buildings and Nuclear Power Plants (Structure-Soil-Structure Interaction), Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions</td>
<td>2588</td>
</tr>
<tr>
<td>510.15.7 Multiple Small Modular Reactors (Structure-Soil-Structure Interaction), Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 3C Seismic Motions</td>
<td>2588</td>
</tr>
<tr>
<td>511 ASCE-4, Chapter on Nonlinear ESSI analysis</td>
<td>2589</td>
</tr>
<tr>
<td>511.1 Motivation: Modeling and Simulation of Earthquake Soil Structure Interaction</td>
<td>2590</td>
</tr>
<tr>
<td>511.2 Introduction</td>
<td>2593</td>
</tr>
<tr>
<td>511.3 Seismic Energy Input and Dissipation</td>
<td>2594</td>
</tr>
<tr>
<td>511.3.1 Seismic Energy Input</td>
<td>2594</td>
</tr>
<tr>
<td>511.3.2 Seismic Energy Dissipation</td>
<td>2594</td>
</tr>
<tr>
<td>511.3.2.1 Seismic Energy Dissipation, Wave Reflection and Wave Radiation</td>
<td>2594</td>
</tr>
<tr>
<td>511.3.2.2 Seismic Energy Dissipation, Viscous Coupling</td>
<td>2594</td>
</tr>
<tr>
<td>Section</td>
<td>Subsection</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>511.3.2.3</td>
<td>Seismic Energy Dissipation, Material Inelasticity</td>
</tr>
<tr>
<td>511.3.2.4</td>
<td>Seismic Energy Dissipation, Numerical, Algorithmic Positive and Negative Damping</td>
</tr>
<tr>
<td>511.4</td>
<td>Modeling: Seismic Motions</td>
</tr>
<tr>
<td>511.4.1</td>
<td>Seismic Motions: Available Data</td>
</tr>
<tr>
<td>511.4.2</td>
<td>Seismic Motion Development</td>
</tr>
<tr>
<td>511.4.2.1</td>
<td>Seismic Motions from Empirical Models</td>
</tr>
<tr>
<td>511.4.2.2</td>
<td>Seismic Motions from Geophysical Models</td>
</tr>
<tr>
<td>511.4.2.3</td>
<td>Seismic Motions from 3D/3C Analytic Models</td>
</tr>
<tr>
<td>511.4.3</td>
<td>6C vs 3C vs 3×1C vs 1C Seismic Motions</td>
</tr>
<tr>
<td>511.4.4</td>
<td>Incoherent Seismic Motions</td>
</tr>
<tr>
<td>511.4.5</td>
<td>Seismic Motion Input into FEM Models</td>
</tr>
<tr>
<td>511.5</td>
<td>Modeling: Inelastic, Nonlinear Material Modeling for Solids and Structures</td>
</tr>
<tr>
<td>511.5.1</td>
<td>Inelastic Material Modeling of Rock</td>
</tr>
<tr>
<td>511.5.1.1</td>
<td>Calibration of Inelastic Material Model Parameters for Rock</td>
</tr>
<tr>
<td>511.5.2</td>
<td>Inelastic Material Modeling of Soil</td>
</tr>
<tr>
<td>511.5.2.1</td>
<td>Dry Soil</td>
</tr>
<tr>
<td>511.5.2.2</td>
<td>Fully Saturated Soil</td>
</tr>
<tr>
<td>511.5.2.3</td>
<td>Partially Saturated, Unsaturated Soil</td>
</tr>
<tr>
<td>511.5.2.4</td>
<td>Calibration of Inelastic Material Model Parameters for Soil</td>
</tr>
<tr>
<td>511.5.3</td>
<td>Inelastic Material Modeling of Steel</td>
</tr>
<tr>
<td>511.5.3.1</td>
<td>Calibration of Inelastic Material Model Parameters for Steel</td>
</tr>
<tr>
<td>511.5.4</td>
<td>Inelastic Material Modeling of Concrete</td>
</tr>
<tr>
<td>511.5.4.1</td>
<td>Calibration of Inelastic Material Model Parameters for Concrete</td>
</tr>
<tr>
<td>511.6</td>
<td>Modeling: Inelastic, Nonlinear Material Modeling for Contacts, Interfaces, and Joints</td>
</tr>
<tr>
<td>511.6.1</td>
<td>Material Modeling of Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)</td>
</tr>
<tr>
<td>511.6.1.1</td>
<td>Calibration of Inelastic Material Model Parameters for Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)</td>
</tr>
<tr>
<td>511.6.2</td>
<td>Material Modeling of Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)</td>
</tr>
<tr>
<td>511.6.2.1</td>
<td>Calibration of Inelastic Material Model Parameters for Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)</td>
</tr>
</tbody>
</table>
511.7 Modeling: Buoyancy ........................................... 2599
511.8 Modeling: Base Isolator and Base Dissipator Systems .......... 2600
  511.8.1 Base Isolator Systems .................................. 2600
  511.8.1.1 Calibration of Elastic/Inelastic Material Model Parameters for Base Isolator Systems ........................................... 2600
  511.8.2 Base Dissipator Systems .................................. 2600
  511.8.2.1 Calibration of Elastic/Inelastic Material Model Parameters for Base Dissipator Systems ........................................... 2600
511.9 Modelling: Finite Element System ........................... 2601
  511.9.1 Mass Matrix ........................................... 2601
  511.9.1.1 Consistent Mass Matrix ............................. 2601
  511.9.1.2 Lumped Mass Matrix ............................. 2601
  511.9.2 Viscous Damping Matrix ............................. 2601
    511.9.2.1 Rayleigh Damping ............................. 2601
    511.9.2.2 Caughey Damping ............................. 2601
  511.9.3 Stiffness Matrix ..................................... 2601
    511.9.3.1 Tangent Stiffness Matrix ..................... 2601
    511.9.3.2 Consistent Stiffness Matrix ................... 2601
511.10 Modeling: Solid, Structure – Fluid Interaction Modeling .... 2602
511.11 Simulation: Nonlinear Finite Elements ....................... 2603
  511.11.1 Time Marching Algorithms for Solution of Nonlinear Equations of Motion .................. 2603
    511.11.1.1 Newmark Algorithm ........................... 2603
    511.11.1.2 Hilber Hughes Taylor $\alpha$ Algorithm ...... 2603
  511.11.2 Solution of Elastic-Plastic Constitutive Equations ........ 2603
    511.11.2.1 Explicit Integration of Elastic-Plastic Constitutive Equations ........ 2603
    511.11.2.2 Implicit Integration of Elastic-Plastic Constitutive Equations ........ 2603
511.12 Modelling Guide for ESSI .................................. 2604
  511.12.1 Buildings and NPPs on Shallow Foundations, Models .... 2604
  511.12.2 Buildings and NPPs on Deeply Embedded Foundation (SMRs), Models .................. 2604
  511.12.3 Buildings and NPPs on Piles and Pile Group Foundations, Models .................. 2604
  511.12.4 Structure – Soil – Structure Interaction, Models ........ 2604
511.13 Practical Steps for Inelastic ESSI Analysis ................... 2605
  511.13.1 Model Development for ESSI ........................... 2605
  511.13.2 Earthquake Soil Structure Interaction: Model Analysis .... 2605
511.13.3 Earthquake Soil Structure Interaction: Results Postprocessing

511.14 Quality Assurance Procedures for ESSI Modeling and Simulation

511.14.1 Verification

511.14.2 Validation

511.15 Practical Examples, Nonlinear, Inelastic ESSI

511.15.1 Nuclear Power Plant, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.2 Nuclear Power Plant on Piles, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.3 Nuclear Power Plant, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 6C/3C/3×1C/1C Seismic Motions

511.15.4 Small Modular Reactor, Deeply Embedded, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.5 Small Modular Reactor, Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil (Cyclic Mobility and Liquefaction), Inelastic Saturated Contact/Interface (Buoyant Pressures), 6C/3C/3×1C/1C Seismic Motions

511.15.6 Multiple Buildings and Nuclear Power Plants (Structure-Soil-Structure Interaction), Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.7 Multiple Small Modular Reactors (Structure-Soil-Structure Interaction), Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 3C Seismic Motions

512 Earthquake-Soil-Structure Interaction, Core Functionality

512.1 Core Functionality for ESSI Analysis of Nuclear Installations

512.2 Model Setup

512.3 Linear Elastic Modeling

512.4 Nonlinear/Inelastic Modeling

512.5 Model Domain

512.5.1 Nodes

512.5.2 Boundary Conditions
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>512.5.3</td>
<td>Static Acceleration Field</td>
<td>2611</td>
</tr>
<tr>
<td>512.5.4</td>
<td>Dynamic Acceleration Field, Earthquake</td>
<td>2611</td>
</tr>
<tr>
<td>512.5.5</td>
<td>Super Element</td>
<td>2611</td>
</tr>
<tr>
<td>512.6</td>
<td>Structural Modeling</td>
<td>2612</td>
</tr>
<tr>
<td>512.6.1</td>
<td>Truss</td>
<td>2612</td>
</tr>
<tr>
<td>512.6.2</td>
<td>Beam</td>
<td>2612</td>
</tr>
<tr>
<td>512.6.3</td>
<td>Shell</td>
<td>2613</td>
</tr>
<tr>
<td>512.7</td>
<td>Solid Modeling</td>
<td>2613</td>
</tr>
<tr>
<td>512.7.1</td>
<td>Solid Brick</td>
<td>2613</td>
</tr>
<tr>
<td>512.7.2</td>
<td>Contact, Interfaces, Joints</td>
<td>2613</td>
</tr>
<tr>
<td>512.8</td>
<td>Core Material Modeling Parameters for Soil, Rock, Concrete, and Steel</td>
<td>2614</td>
</tr>
<tr>
<td>512.8.1</td>
<td>Linear and Nonlinear Elastic Soil, Rock, Concrete, and Steel Modeling</td>
<td>2614</td>
</tr>
<tr>
<td>512.8.2</td>
<td>Inelastic/Nonlinear Soil Modeling</td>
<td>2614</td>
</tr>
<tr>
<td>512.8.3</td>
<td>Inelastic/Nonlinear Rock Modeling</td>
<td>2615</td>
</tr>
<tr>
<td>512.8.4</td>
<td>Inelastic/Nonlinear Concrete Modeling</td>
<td>2615</td>
</tr>
<tr>
<td>512.8.5</td>
<td>Inelastic/Nonlinear Steel Modeling</td>
<td>2615</td>
</tr>
<tr>
<td>512.9</td>
<td>Core Material Modeling Parameters for Contacts, Interfaces and Joints</td>
<td>2615</td>
</tr>
<tr>
<td>512.9.1</td>
<td>Mass Concrete Against Silt, Sand, Gravel and Clay</td>
<td>2616</td>
</tr>
<tr>
<td>512.9.2</td>
<td>Steel Sheet Against Sand, Gravel and Rockfill</td>
<td>2617</td>
</tr>
<tr>
<td>512.9.3</td>
<td>Formed Concrete Against Sand, Gravel and Rockfill</td>
<td>2618</td>
</tr>
<tr>
<td>512.10</td>
<td>Earthquake Motion Modeling</td>
<td>2619</td>
</tr>
<tr>
<td>512.10.1</td>
<td>One Component (1C) Seismic Motions Defined at Surface or at Depth</td>
<td>2619</td>
</tr>
<tr>
<td>512.10.2</td>
<td>$3 \times 1C$ Seismic Motions Defined at Surface or at Depth</td>
<td>2620</td>
</tr>
<tr>
<td>512.10.3</td>
<td>Seismic Motions Imposed at Model Base</td>
<td>2620</td>
</tr>
<tr>
<td>512.10.4</td>
<td>Eigen Analysis</td>
<td>2620</td>
</tr>
<tr>
<td>512.11</td>
<td>Core Modeling and Simulation Commands: Simulation Parameters</td>
<td>2620</td>
</tr>
</tbody>
</table>

600 References

700 Appendix

701 Useful Formulae

(1985-1989-1993-2021-)

701.1 Chapter Summary and Highlights
701.2 Stress and Strain ........................................ 2692
  701.2.1 Stress ........................................... 2692
  701.2.2 Strain ........................................... 2695
701.3 Derivatives of Stress Invariants ....................... 2697

702 The \texttt{nDarray} Programming Tool


702.1 Chapter Summary and Highlights ....................... 2702
702.2 Introduction ........................................... 2702
702.3 \texttt{nDarray} Programming Tool ....................... 2703
  702.3.1 Introduction to the \texttt{nDarray} Programming Tool .... 2703
  702.3.2 Abstraction Levels ................................ 2704
    702.3.2.1 \texttt{nDarray\_rep} class ....................... 2705
    702.3.2.2 \texttt{nDarray} class ......................... 2705
    702.3.2.3 Matrix and Vector Classes .................... 2706
    702.3.2.4 Tensor Class .................................. 2706
  702.4 Finite Element Classes ................................ 2708
    702.4.1 Stress, Strain and Elastoplastic State Classes ...... 2708
    702.4.2 Material Model Classes .......................... 2709
    702.4.3 Stiffness Matrix Class ........................... 2710
  702.5 Examples ............................................ 2711
    702.5.1 Tensor Examples ................................. 2711
    702.5.2 Fourth Order Isotropic Tensors .................... 2712
    702.5.3 Elastic Isotropic Stiffness and Compliance Tensors .... 2713
    702.5.4 Second Derivative of $\theta$ Stress Invariant ....... 2714
    702.5.5 Application to Computations in Elastoplasticity .... 2715
    702.5.6 Stiffness Matrix Example ........................ 2716
  702.6 Performance Issues ................................... 2717
  702.7 Summary and Future Directions ....................... 2717

703 Closed Form Gradients to the Plastic Potential Function

(1993-1994-) .................................................. 2718

703.1 Chapter Summary and Highlights ....................... 2719

704 Hyperelasticity, Detailed Derivations

(1995-1996-) .................................................. 2726

704.1 Chapter Summary and Highlights ....................... 2727
704.2 Simo–Serrin’s Formula
704.3 Derivation of $\partial^2 \omega W / (\partial C_{1,1} \partial C_{K,L})$
704.4 Derivation of $\partial^2 \omega W / (\partial C_{1,1} \partial C_{K,L})$
704.5 Derivation of $w_A$
704.6 Derivation of $Y_{AB}$

705 Body and Surface Wave Analytic Solutions


705.1 3D Seismic Wave Field: Analytic Solution

705.1.1 Analytic solution

705.1.1.1 Wave equations for body waves
705.1.1.2 Wave equations for surface waves

705.2 Matlab code – body wave solution
705.3 Matlab code – surface wave solution
705.4 Matlab code – Ricker wavelet as an input motion

706 Body and Surface Wave Numerical Modeling


706.1 Integral equations

706.1.1 fk3.0 package

706.1.1.1 fk and ‘sample_input’
706.1.1.2 fk.pl
706.1.1.3 syn
706.1.1.4 plot.py

706.1.2 3D seismic wave field generation using integral equation

706.1.2.1 Case 1: strike-slip fault / single layer ground
706.1.2.2 Case 2: dip-slip fault / single layer ground
706.1.2.3 Case 3: normal fault / single layer ground
706.1.2.4 Case 4: normal fault / layered ground
706.1.2.5 Case 5: Northridge earthquake / layered ground

707 Real-ESSI Illustrative Examples


707.1 Elastic Beam Element Under Static Loading
707.2 Elastic Beam Element under Dynamic Loading
707.3 Cantilever, 5 Elastic Beam Elements
707.4 Cantilever, One 27 Node Brick Element, Dynamic Loading
707.5 Simulate Cantilever Using Five 27 Node Brick Elements .......................... 2827
707.6 Elastic Beam Element under Dynamic Loading with concentrated mass .......... 2833
707.7 Elastic Beam, 27 Node Brick Model With Concentrated Mass ....................... 2837
707.8 Elastic Beam Element, Dynamic Loading, Viscous (Rayleigh/Caughey) and Numerical (Newmark/HHT) Damping ................................. 2844
707.9 Elastic Beam Element for a Simple Frame Structure ................................. 2851
707.10 27NodeBrick Cantilever Beam, Static Load .......................................... 2853
707.11 4NodeANDER Cantilever Beam, Force Perpendicular to Plane .................. 2856
707.12 4NodeANDER Cantilever Beams, In-Plane Force .................................. 2858
707.13 27NodeBrick Cantilever Beams, Dynamic Input ....................................... 2860
707.14 4NodeANDER Square Plate, Four Edges Clamped ................................... 2864
707.15 One Dimensional DRM Model ............................................................... 2867
707.16 Three Dimensional DRM Model ............................................................. 2872
707.17 ShearBeam Element, Pisano Material ...................................................... 2875
707.18 8NodeBrickLT Element, Drucker-Prager Material, Armstrong-Frederick Rotational Kinematic Hardening ......................................................... 2878
707.19 Contact Element Under Static Loading .................................................... 2883
707.20 Four Bar Contact Problem With Normal and Shear Force Under Monotonic Loading ................................................................. 2885
707.21 3-D Truss example with normal confinement and Shear Loading ................. 2889
707.22 Six Solid Blocks Example With Contact ................................................... 2892
707.23 Pure shear model for G/Gmax plot ........................................................... 2899
707.24 Multi-yield-surface von-Mises for G/Gmax plot ........................................ 2903
707.25 Multi-yield-surface Drucker-Prager for G/Gmax plot ................................... 2906
708 Brief History of the Real-ESSI Simulator Development ................................. 2909
709 Work Organization ....................................................................................... 2912
709.1 Communication ....................................................................................... 2913
709.2 Writing (Notes, Code, &c.) Version Control ................................................. 2913
  709.2.1 Source Code ......................................................................................... 2913
  709.2.2 Lecture Notes ....................................................................................... 2914
  709.2.3 Bibliography ........................................................................................ 2914
709.3 Backup ................................................................. 2915
709.4 Calendar ............................................................... 2915
709.5 Useful Programs and Scripts ........................................ 2915
  709.5.1 Backup Scripts ................................................... 2915
  709.5.2 Domain Reduction Method Processing Programs and Scripts ................................................... 2915
  709.5.3 Pre Processing Programs and Scripts ................................................... 2915
  709.5.4 Post Processing Programs and Scripts ................................................... 2915
  709.5.5 Parallel Computer Architecture ................................................... 2915
710 Collected Bibliography ................................................ 2916
Part 100

Theoretical and Computational Formulations
Chapter 101

Introduction

101.1 Chapter Summary and Highlights

101.2 Specialization to Computational Mechanics

In this section we start from general mechanics and specialize our interest toward the field of computational mechanics (this is based on great lecture notes by Prof. Carlos Felippa (Felippa, 1993)):

101.2.1 Mechanics

- Theoretical
- Applied
- Computational
  - Nanomechanics
  - Micromechanics
  - Continuum Mechanics
    * Solids and Structures
    * Fluids
    * Multiphysics
  - Systems
- Experimental

101.2.2 Continuum Mechanics

- Statics
  - Time invariant
  - Transient (quasi-statics)
- Dynamics

101.2.3 Statics and Dynamics

- Linear
- Nonlinear
- Elastic
- Inelastic

101.2.4 Discretization Methods

- Finite Element Method (FEM)
  - FEM Formulation
    * Displacement
    * Equilibrium
    * Mixed
    * Hybrid
  - FEM Solution
    * Stiffness
    * Flexibility
    * Mixed
- Boundary Element Method
- Finite Difference Method
- Finite Volume Method
- Spectral Method
- Mesh-Free Method

101.2.5 The Solution Morass

A system of 1000 linear equations has one solution.

A system of 1000 cubic equations has $3^{1000} \approx 10^{477}$ solutions.

It is worth putting this number in perspective: number of atoms in the earth is about $10^{50}$, and a number of atoms in the universe is about $10^{78}$ (Niemunis, 2015 –).

Solution: Continuation or Incremental analysis!
101.2.6 Smooth Nonlinearities

- Finite deflections
- Nonlinear elasticity
- Follower forces

101.2.7 Rough Nonlinearities

- Elasto-plasticity
- Contact/Interface/Joint
- Interface/joint Friction

101.3 Tour of Computational Mechanics

In this section we describe various examples of equilibrium path and set up basic terminology.

101.3.1 Equilibrium Path

101.3.2 Special Equilibrium Points

101.3.2.1 Critical Points
- Limit Points

- Bifurcation Points

101.3.2.2 Turning Points

101.3.2.3 Failure Points

101.3.3 Generalized Response

101.3.4 Sources of Nonlinearities
- Tonti Diagrams

101.3.5 Simulation Process: Loading Stages, Increments and Iterations
Figure 101.1: Nonlinear analysis loading stages, loading increments, equilibrium iterations.
Chapter 102

Finite Elements Formulation

(In collaboration with Dr. Zhao Cheng, Dr. Nima Tafazzoli, Prof. José Abell, Dr. Yuan Feng, Dr. Han Yang)
102.1 Chapter Summary and Highlights

This section uses basic principles of mechanics to derive finite element equations. We start with general setup, assuming large deformation in section 102.2, and then proceed to develop finite element formulation for small deformations in section 102.3 on page 97. Further investigation of large deformation formulation is given in chapter 106 on page 451.

102.2 Formulation of the Continuum Mechanics Incremental Equations of Motion

This section follows Bathe (1982), Felippa (1989) and Felippa (1993).

Assume that a 3D solid is analyzed in a fixed Cartesian coordinate system, Figure (102.1). Also, assume that the solid can undergo large displacements and rotations, large strains, and nonlinear or inelastic constitutive response. The main aim is to evaluate the equilibrium of solid at discrete times $0, \Delta t, 2\Delta t, \ldots$, where $\Delta t$ is an increment in time. To do that, a continuation strategy is used. That is, assume that the solution for all the variables (generalized displacements, strain, stress, forces, etc.) is available, was solved for, for all time steps from 0 to time $t$. Solution for the next time step $t + \Delta t$ will be based on already obtained solution from the previous time step, at time $t$, (Felippa, 1993). This approach will be applied for each time step, repetitively until the solution for all time steps is obtained.

In following all parts of the solid, as they undergo displacements and rotations, from the original configuration to the final configuration, adopted is a Lagrangian (or material) formulation of the problem. This approach is contrasting Eulerian (or spatial) formulation, usually used in the analysis of fluid mechanics problems.

In the Lagrangian incremental analysis approach the equilibrium of the solid at time $t + \Delta t$ is expressed using the principle of virtual displacements. Using tensorial notation\footnote{Einstein’s summation rule is implied unless stated differently, all lower case indices $\left( i, j, p, q, m, n, o, r, s, t, \ldots \right)$ can have values of 1, 2, 3, and values for capital letter indices will be specified where need be.} this principle requires that:

$$\int_{t + \Delta t}^{t + 2\Delta t} \sigma_{ij} \frac{\delta}{\delta t} \epsilon_{ij} dV = R$$

where the $\sigma_{ij}$ are Cartesian components of the Cauchy stress tensor, see section 701.2.1 on page 2692, the $\epsilon_{ij}$ are the Cartesian components of an infinitesimal strain tensor, see section 701.2.2 on
\[ \delta_{t+\Delta t} \varepsilon_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial t+\Delta t, x_j} + \frac{\partial u_j}{\partial t+\Delta t, x_i} \right) = \frac{1}{2} \left( \frac{\partial \delta u_i}{\partial t+\Delta t, x_j} + \frac{\partial \delta u_j}{\partial t+\Delta t, x_i} \right) \]  

It should be noted that Cauchy stresses are "body forces per unit area" in the configuration at time \( t + \Delta t \), and the infinitesimal strain components are also referred to this as yet unknown configuration.

The right hand side of equation (102.1), i.e. \( t+\Delta t R \) is the virtual work performed when the solid is subjected to a virtual displacement at time \( t + \Delta t \):

\[ t+\Delta t R = \int_{t+\Delta t V} \left( f_i^B(t+\Delta t) - \rho u_i^{t+\Delta t} \right) \delta u_i^{t+\Delta t} + \int_{t+\Delta t S} f_i^S \delta u_i^{t+\Delta t} + \int_{t+\Delta t dS} \]  

where \( t+\Delta t f_i^B \) and \( t+\Delta t f_i^S \) are the components of the externally applied body and surface force vectors, respectively, and \(-\rho u_i^{t+\Delta t}\) is the inertial body force that is present if accelerations are present \(^2\), \( \delta u_i \) is the \( i \)th component of the virtual displacement vector.

The main problem in applying equation (102.1) is that the configuration of the solid at a time \( t + \Delta t \) is unknown. The continuous change in the configuration, deformation of the solid needs to be properly

\(^2\) This is based on D’Alembert’s principle (D’Alembert, 1758).
modeled. As an example, consider, for example, Cauchy stress at time $t + \Delta t$. This stress cannot be obtained by adding to the Cauchy stresses at time $t$, a stress increment that is due only to material deformation. The reason is that material might rotate, and stress state is a function of tractions (loads) and size and orientation of differentially small faces on which stress components act. For material only nonlinear analysis, the of large displacements, large rotations and large strain will be neglected. Large displacements, rotations and large strains will be addressed in more detail in Chapter 106 on page 451.

The continuous change in the configuration of the solid is dealt with by using appropriate stress and strain measures and constitutive relations. When solving the general problem\(^3\) one possible approach is given in Simo (1988). The previous discussion was oriented toward small deformation, small-displacement analysis leading to the use of Cauchy stress tensor $\sigma_{ij}$ and small strain tensor $\epsilon_{ij}$.

In the following, covered briefly are other stress and strain measures particularly useful in large strain and large displacement analysis. More detailed description of large displacements, large rotations and large strains problems is addressed in Chapter 106 on page 451.

The solution is sought for equation 102.1, which expresses the equilibrium and compatibility requirements of the general solid considered in the configuration corresponding to time $t + \Delta t$. The nonlinear or inelastic behavior of material enters equation 102.1 through the stress-strain constitutive equations. In general, the solid can undergo large displacements, large rotations, large strains, and since constitutive relations are nonlinear, the relation in equation 102.1 cannot be solved directly. However, an approximate solution can be obtained by referring all variables to a previously calculated known equilibrium configuration, and linearizing the resulting equations. Iterations can then improve this solution.

To develop the governing equations for the approximate solution obtained by linearization, recall that the solutions for time $0, \Delta t, 2\Delta t, \ldots, t$ have already been calculated and that the Piola–Kirchhoff stress tensor is energy conjugate to the Green–Lagrange strain tensor:

$$
\int_0^V \psi_{ij} \delta \xi_{ij} 0 dV = \int_0^V \left( \frac{\partial \psi}{\partial \xi} \right)_{0}^{0} \sigma_{mn} \delta \epsilon_{mn} 0 dV = \int_0^V \frac{\partial \psi}{\partial \epsilon} \sigma_{mn} \delta \xi_{mn} 0 dV
$$

(102.3)

since

$$
\psi_{ij} \delta \xi_{ij} = \delta_{km}
$$

and

\(^3\)That is, large displacements, large rotations, large deformations and material nonlinear and/or inelastic.
\[ 0 \rho^0 dV = t \rho^t dV \]

one obtains

\[ \int_{V}^t s_{ij}^t \delta_{t_{ij}}^t 0 dV = \int_{V}^t \sigma_{mn}^t \delta_{t_{mn}}^t t dV \tag{102.4} \]

where 2nd Piola–Kirchhoff stress tensor is defined as:

\[ t^0 s_{ij} = \frac{0 \rho}{t \rho} x_{i,m} t x_{j,n} \sigma_{mn}^0 \]

(102.5)

and \( t^0 x_{j,n} = \frac{\partial^t x_{j}}{\partial x_{m}} \), and \( \frac{0 \rho}{t \rho} \) represents the ratio of the mass density at time 0 and time \( t \), and the Green–Lagrange strain is defined as:

\[ s_{ij}^t = \frac{1}{2} \left( 0^t u_{i,j} + 0^t u_{j,i} + 0^t u_{k,i}^t t u_{k,j} \right) \tag{102.6} \]

By employing equation 102.4 stresses and strains are referenced to the known equilibrium configuration. The choice lies between two formulations, named (a) total Lagrangian and (b) updated Lagrangian formulations.

For the total Lagrangian formulations, all static and kinematic variables are referenced to the initial configuration at time 0. On the other hand, for the updated Lagrangian formulation, all static and kinematic variables are referenced to the previous step equilibrium configuration at time \( t \). Both the total Lagrangian and updated Lagrangian formulations include all kinematic nonlinear effects due to large displacement, large rotations, and large strains. Whether the large strain behavior is modeled appropriately depends on the constitutive relations specified. The only advantage of using one over the other formulation lies in numerical efficiency.

Using equation 102.4 in the total Lagrangian formulation, considered is this equation:

\[ \int_{V}^t s_{ij}^t \delta_{t_{ij}}^t 0 dV = t + \Delta t R \tag{102.7} \]

while in the updated Lagrangian formulation considered is this equation:

\[ \int_{V}^t s_{ij}^t \delta_{t_{ij}}^t t dV = t + \Delta t R \tag{102.8} \]
where \( t + \Delta t \mathcal{R} \) is the external virtual work as defined in equation ??.

Approximate solution to the equation 102.7 and equation 102.8 can be obtained by linearization. Comparison of the total Lagrangian and updated Lagrangian formulations reveal that they are quite similar, with the difference in the choice of different reference configurations for kinematic and static variables. If in the numerical solution the appropriate constitutive tensors are employed, identical results should be obtained.

### 102.3 Finite Element Discretization

Consider the equilibrium of a general three–dimensional solid such as in Figure (102.2) (Bathe, 1996). The external forces acting on a solid are surface tractions \( f^S_i \) and body forces \( f^B_i \). Displacements are \( u_i \) and strain tensor\(^4\) is \( \epsilon_{ij} \) and the stress tensor corresponding to strain tensor is \( \sigma_{ij} \).

\[ \sigma_{ij,j} = f_i - \rho \ddot{u}_i \quad (102.9) \]

where \( \sigma_{ij,j} \) is a small deformation (Cauchy) stress tensor, \( f_i \) are external (body \( f^B_i \) and surface \( f^S_i \)) forces, \( \rho \) is material density and \( \ddot{u}_i \) are accelerations. Inertial forces \( \rho \ddot{u}_i \) follow from D’Alembert’s principle.

\(^4\) small strain tensor as defined in equation: \( \epsilon_{ij} = \frac{1}{2} (u_{i,j} + u_{j,i}) \).
The above equation can be premultiplied with virtual displacements $\delta u_i$ and then integrated by parts to obtain the weak form, as further elaborated below.

For a given solid, loaded with external forces, with prescribed supports solution for displacements, strains and stresses are sought. The principle of virtual displacements (PVD) can be used to find a solution. Using PVD, equilibrium is achieved if the total internal virtual work is equal to the total external virtual work, for any compatible, small virtual displacements, that satisfy the essential boundary conditions.

Mathematically this is expressed using equation 102.10 for the solid at time $t + \Delta t$. Since the incremental approach is used, a time dimension is dropped so that all the equations are applied for the given increment\(^5\), at time $t + \Delta t$. The equation is now, using tensorial notation\(^6\):

$$
\int_V \sigma_{ij} \delta \epsilon_{ij} \, dV = \int_V \left(f^B_i - \rho \ddot{u}_i\right) \delta u_i \, dV + \int_S f^S_i \delta u_i \, dS
$$

\[(102.10)\]

The internal work given on the left side of (102.10) is equal to the actual stresses $\sigma_{ij}$ going through the virtual strains $\delta \epsilon_{ij}$ that corresponds to the imposed virtual displacements. The external work is on the right side of (102.10) and is equal to the actual surface forces $f^S_i$ and body forces $f^B_i - \rho \ddot{u}_i$ going through the virtual displacements $\delta u_i$.

It is noted virtual strains used in equation 102.10 correspond to the imposed virtual displacements that can be represented by any compatible set of displacements that satisfy the geometric boundary conditions. The equation 102.10 is an expression of equilibrium, and for different virtual displacements, correspondingly, different equations of equilibrium are obtained. Equation 102.10 also contains the compatibility and constitutive requirements. Displacements should be continuous and compatible and should satisfy the displacement boundary conditions, and the stresses should be evaluated from the strains using appropriate constitutive relations. Thus, the principle of virtual displacements contains all requirements that need to be fulfilled to analyze a problem in solid and structural mechanics. The principle of virtual displacements can be directly related to the principle that the total potential $\Pi$ of the system must be stationary.

In the finite element analysis, approximation for the solid in Figure 102.2 is done by creating an assemblage of discrete finite elements with the elements connected at nodal points. The displacements measured in a local coordinate system $r_1$, $r_2$ and $r_3$ within each element are assumed to be a function

---

\(^5\) $t + \Delta t$ will be dropped from now one in this chapter.

\(^6\) Einstein’s summation rule is implied unless stated differently, all lower case indices $(i, j, p, q, m, n, o, r, s, t, \ldots)$ can have values of 1, 2, 3, and values for capital letter indices will be specified where need be.
of the displacements at the \( N \) finite element nodal points:

\[
    u_i \approx \hat{u}_a = H_I \bar{u}_{Ia}
\]  

(102.11)

where \( I = 1, 2, 3, \ldots, n \) and \( n \) is number of nodes in a specific element, \( a = 1, 2, 3 \) represents a number of dimensions (can be 1 or 2 or 3). Real displacement field \( u_i \) is approximated with approximate displacement field \( \hat{u}_a \), and \( H_I \) represent displacement interpolation functions, \( \bar{u}_{Ia} \) is the tensor of global approximate generalized displacement components at all element nodes. The term generalized displacements mean that both translations, rotations, or any other nodal unknown are modeled independently. Here specifically, only translational degrees of freedom are considered. The strain tensor is defined as:

\[
    \epsilon_{ab} = \frac{1}{2} (u_{a,b} + u_{b,a})
\]  

(102.12)

and by using equation 102.11, approximate strain tensor can be defined as:

\[
    \epsilon_{ab} \approx \hat{\epsilon}_{ab} = \frac{1}{2} (\hat{u}_{a,b} + \hat{u}_{b,a}) = \\
    = \frac{1}{2} ( (H_I \bar{u}_{Ia})_{,b} + (H_I \bar{u}_{Ib})_{,a} ) = \\
    = \frac{1}{2} ( (H_{I,b} \bar{u}_{Ia}) + (H_{I,a} \bar{u}_{Ib}) )
\]  

(102.13)

The most general stress-strain relationship\(^7\) for an isotropic material is:

\[
    \hat{\sigma}_{ab} = E_{abcd} (\hat{\epsilon}_{cd} - \epsilon_{cd}^0) + \sigma_{ab}^0
\]  

(102.14)

where \( \hat{\sigma}_{ab} \) is the approximate Cauchy stress tensor, \( E_{abcd} \) is the constitutive tensor\(^8\), \( \hat{\epsilon}_{cd} \) is the infinitesimal approximate strain tensor, \( \epsilon_{cd}^0 \) is the infinitesimal initial strain tensor and \( \sigma_{ab}^0 \) is the initial Cauchy stress tensor.

Using the assumption of the displacements within each finite element, as expressed in equation 102.11, equilibrium equations that correspond to nodal point displacements of the assemblage of finite elements can be derived. Equation 102.10 can be rewritten as a sum\(^9\) of integrations over the volume and areas of all finite elements:

\[
    \bigcup_m \int_{V_m} \hat{\sigma}_{ab} \delta \hat{e}_{ab} \, dV_m = \bigcup_m \int_{V_m} (f_{a}^B - \rho \ddot{u}_a) \delta u_a \, dV_m + \bigcup_m \int_{S_m} f_a^S \delta u_a^S \, dS_m
\]

\(^7\)In terms of exact stress and strain fields, but it holds for approximate fields as well.
\(^8\)This tensor can be elastic or elastoplastic constitutive tensor.
\(^9\)Or, more correctly as a union \( \bigcup_m \) since we are integrating over the union of elements.
where \( m = 1, 2, 3, \ldots, k \) and \( k \) is the number of elements. It is important to note that the integrations in \( (??) \) are performed over the element volumes and surfaces, and that for convenience different element coordinate systems may be used in the calculations. If we substitute equations \( 102.11, 102.12, 102.13 \) and \( 102.14 \) in \( (??) \), it follows:

\[
\bigcup_m \int_{V_m} \left( E_{abcd} \left( \epsilon_{cd}^0 + \sigma_{ab}^0 \right) - \frac{1}{2} \left( H_{I,b} \delta \bar{u}_{Ia} + H_{I,a} \delta \bar{u}_{ib} \right) \right) \delta \left( \frac{1}{2} \left( H_{I,b} \delta \bar{u}_{Ia} + H_{I,a} \delta \bar{u}_{ib} \right) \right) dV_m = \\
\bigcup_m \int_{V_m} f_a^B \delta \left( H_{I,Ja} \right) dV_m - \bigcup_m \int_{V_m} H_{Ja} \delta \left( H_{I,Ja} \right) dV_m + \bigcup_m \int_{S_m} f_a^S \delta \left( H_{I,Ja} \right) dS_m
\]

(102.15)

or:

\[
\bigcup_m \int_{V_m} \left( E_{abcd} \left( \left( \frac{1}{2} \left( H_{J,d} \delta \bar{u}_{Jc} + H_{J,c} \delta \bar{u}_{Jd} \right) \right) - \epsilon_{cd}^0 + \sigma_{ab}^0 \right) \right) \delta \left( \frac{1}{2} \left( H_{I,b} \delta \bar{u}_{Ia} + H_{I,a} \delta \bar{u}_{ib} \right) \right) dV_m = \\
\bigcup_m \int_{V_m} f_a^B \delta \left( H_{I,Ja} \right) dV_m - \bigcup_m \int_{V_m} H_{Ja} \delta \left( H_{I,Ja} \right) dV_m + \bigcup_m \int_{S_m} f_a^S \delta \left( H_{I,Ja} \right) dS_m
\]

(102.16)

We can observe that \( \delta \) in the previous equations represents a virtual quantity, but the rules for \( \delta \) are quite similar to regular differentiation so that \( \delta \) can enter the brackets and "virtualize" the nodal displacement\(^\text{10}\). It thus follows:

\[
\bigcup_m \int_{V_m} \left( E_{abcd} \left( \left( \frac{1}{2} \left( H_{J,d} \delta \bar{u}_{Jc} + H_{J,c} \delta \bar{u}_{Jd} \right) \right) - \epsilon_{cd}^0 + \sigma_{ab}^0 \right) \right) \left( \frac{1}{2} \left( H_{I,b} \delta \bar{u}_{Ia} + H_{I,a} \delta \bar{u}_{ib} \right) \right) dV_m = \\
\bigcup_m \int_{V_m} f_a^B \delta \left( H_{I,Ja} \right) dV_m - \bigcup_m \int_{V_m} H_{Ja} \delta \left( H_{I,Ja} \right) dV_m + \bigcup_m \int_{S_m} f_a^S \delta \left( H_{I,Ja} \right) dS_m
\]

(102.17)

\(^{10}\)since they are driving variables that define the overall displacement field through interpolation functions
Let us now work out some algebra on the left-hand side of the equation (102.17):

\[ \bigcup_m \int_{V^m} \left( \frac{E_{abcd} (H_{I,d} \bar{u}_{Ja} + H_{J,c} \bar{u}_{jd})}{2} - E_{abcd} \epsilon_{cd}^0 + \sigma_{ab}^0 \right) \left( \frac{H_{I,b} \delta \bar{u}_{Ja} + H_{I,a} \delta \bar{u}_{jb}}{2} \right) dV^m = \]

\[ = \bigcup_m \int_{V^m} f_a^B (H_I \delta \bar{u}_{Ja}) dV^m - \bigcup_m \int_{V^m} H_J \ddot{u}_{Ja} \rho H_I \delta \bar{u}_{Ja} dV^m + \bigcup_m \int_{S^m} f_a^S (H_I \delta \bar{u}_{Ja}) dS^m \]

(102.18)

and further:

\[ \bigcup_m \int_{V^m} \left( \left( \frac{1}{2} (H_{I,d} \bar{u}_{Ja} + H_{J,c} \bar{u}_{jd}) \right) E_{abcd} \left( \frac{1}{2} (H_{I,b} \delta \bar{u}_{Ja} + H_{I,a} \delta \bar{u}_{jb}) \right) \right) dV^m + \]

\[ + \bigcup_m \int_{V^m} \left( -E_{abcd} \epsilon_{cd}^0 \left( \frac{1}{2} (H_{I,b} \delta \bar{u}_{Ja} + H_{I,a} \delta \bar{u}_{jb}) \right) \right) dV^m + \]

\[ + \bigcup_m \int_{V^m} \left( \sigma_{ab}^0 \right) \left( \frac{1}{2} (H_{I,b} \delta \bar{u}_{Ja} + H_{I,a} \delta \bar{u}_{jb}) \right) dV^m = \]

\[ \bigcup_m \int_{V^m} f_a^B (H_I \delta \bar{u}_{Ja}) dV^m \]

\[ - \bigcup_m \int_{V^m} H_J \ddot{u}_{Ja} \rho H_I \delta \bar{u}_{Ja} dV^m + \bigcup_m \int_{S^m} f_a^S (H_I \delta \bar{u}_{Ja}) dS^m \]

(102.19)

Several things should be observed in the equation (102.19). Namely, the first three lines in the equation can be simplified if one takes into account symmetries of $E_{ijkl}$ and $\sigma_{ij}$. In the case of the elastic stiffness tensor $E_{ijkl}$ major and both minor symmetries exist. In the case of the elastoplastic stiffness tensor, such symmetries exist if a flow rule is associated. If the flow rule is non–associated, only minor symmetries exist while major symmetry is destroyed. As a matter of fact, both minor symmetries in $E_{ijkl}$ are the only symmetries needed, and the first line of (102.19) can be rewritten as:

\[^{11}\text{for more on stiffness tensor symmetries, see sections (104.6.1, 104.3 and 104.4)}\]
\[ \bigcup_m \int_{V^m} \left( \left( \frac{1}{2} (H_{J,d} \ddot{u}_{Jc} + H_{J,c} \ddot{u}_{jd}) \right) E_{abcd} \left( \frac{1}{2} (H_{I,b} \delta \ddot{u}_{Ia} + H_{I,a} \delta \ddot{u}_{Ib}) \right) \right) \, dV^m = \]
\[ = \bigcup_m \int_{V^m} (H_{J,d} \ddot{u}_{Jc}) E_{abcd} (H_{I,b} \delta \ddot{u}_{Ia}) \, dV^m = \]
\[ = \bigcup_m \int_{V^m} (H_{I,b} \delta \ddot{u}_{Ia}) E_{abcd} (H_{J,d} \ddot{u}_{Jc}) \, dV^m \]
\[ (102.20) \]

Similar simplifications are possible in the second and third line of the equation (102.19). Namely, in the second line both minor symmetries of \( E_{ijkl} \) can be used so that:

\[ \bigcup_m \int_{V^m} \left( -E_{abcd} \epsilon^0_{cd} \left( \frac{1}{2} (H_{I,b} \delta \ddot{u}_{Ia} + H_{I,a} \delta \ddot{u}_{Ib}) \right) \right) \, dV^m = \]
\[ = \bigcup_m \int_{V^m} \left( -E_{abcd} \epsilon^0_{cd} (H_{I,b} \delta \ddot{u}_{Ia}) \right) \, dV^m \]
\[ (102.21) \]

and the third line can be simplified due to the symmetry in Cauchy stress tensor \( \sigma_{ij} \) as:

\[ \bigcup_m \int_{V^m} \left( \sigma^0_{ab} \left( \frac{1}{2} (H_{I,b} \delta \ddot{u}_{Ia} + H_{I,a} \delta \ddot{u}_{Ib}) \right) \right) \, dV^m = \]
\[ = \bigcup_m \int_{V^m} \left( \sigma^0_{ab} (H_{I,b} \delta \ddot{u}_{Ia}) \right) \, dV^m \]
\[ (102.22) \]

After these simplifications, equation (102.19) looks like this:

\[ \bigcup_m \int_{V^m} (H_{I,b} \delta \ddot{u}_{Ia}) E_{abcd} (H_{J,d} \ddot{u}_{Jc}) \, dV^m + \]
\[ + \bigcup_m \int_{V^m} \left( -E_{abcd} \epsilon^0_{cd} (H_{I,b} \delta \ddot{u}_{Ia}) \right) \, dV^m + \bigcup_m \int_{V^m} \left( \sigma^0_{ab} (H_{I,b} \delta \ddot{u}_{Ia}) \right) \, dV^m = \]
\[ = \bigcup_m \int_{V^m} f^B_a (H_I \delta \ddot{u}_{Ia}) \, dV^m - \bigcup_m \int_{V^m} H_J \ddot{u}_{Ja} \rho I \delta \ddot{u}_{Ia} \, dV^m + \bigcup_m \int_{S^m} f^S_a (H_I \delta \ddot{u}_{Ia}) \, dS^m \]
\[ (102.23) \]
or if unknown nodal accelerations\(^{12}\) \(\ddot{u}_{Jc}\) and displacements \(\bar{u}_{Jc}\) are left on the left hand side and all known quantities are moved to the right hand side:

\[
\bigcup_m \int_{V_m} H_J \delta_{ac} \ddot{u}_{Jc} \rho H_I \delta \bar{u}_{Ia} \, dV_m + \bigcup_m \int_{V_m} (H_{I,b} \, \delta \bar{u}_{Ia}) \, E_{abcd} \, (H_{J,d} \, \ddot{u}_{Jc}) \, dV_m = \\
= \bigcup_m \int_{V_m} f_a^B \, (H_I \delta \bar{u}_{Ia}) \, dV_m + \bigcup_m \int_{S_m} f_a^S \, (H_I \delta \bar{u}_{Ia}) \, dS_m + \\
+ \bigcup_m \int_{V_m} (E_{abcd} \, \epsilon^0_{cd} \, (H_{I,b} \, \delta \bar{u}_{Ia})) \, dV_m - \bigcup_m \int_{V_m} (\sigma^0_{ab} \, (H_{I,b} \, \delta \bar{u}_{Ia})) \, dV_m
\]

(102.24)

To obtain the equation for the unknown nodal generalized displacements from equation 102.24, invoke the virtual displacement theorem. This theorem states that virtual displacements are any, non-zero, kinematically admissible displacements. In that case, we can factor out nodal virtual displacements \(\delta \bar{u}_{Ia}\) so that equation 102.24 becomes:

\[
\left[ \bigcup_m \int_{V_m} H_J \delta_{ac} \ddot{u}_{Jc} \rho H_I \, dV_m + \bigcup_m \int_{V_m} (H_{I,b} \, \delta \bar{u}_{Ia}) \, E_{abcd} \, (H_{J,d} \, \ddot{u}_{Jc}) \, dV_m \right] \delta \bar{u}_{Ia} = \\
= \bigcup_m \left[ \int_{V_m} f_a^B \, H_I \, dV_m \right] \delta \bar{u}_{Ia} + \bigcup_m \left[ \int_{S_m} f_a^S \, H_I \, dS_m \right] \delta \bar{u}_{Ia} + \\
+ \bigcup_m \left[ \int_{V_m} (E_{abcd} \, \epsilon^0_{cd} \, H_{I,b}) \, dV_m \right] \delta \bar{u}_{Ia} - \bigcup_m \left[ \int_{V_m} (\sigma^0_{ab} \, H_{I,b}) \, dV_m \right] \delta \bar{u}_{Ia}
\]

(102.25)

and now just cancel \(\delta \bar{u}_{Ia}\) on both sides:

\(^{12}\)It is noted that \(\ddot{u}_{Jc} = \delta_{ac} \ddot{u}_{Ja}\) relationship was used here, where \(\delta_{ac}\) is the Kronecker delta.
\[
\bigcup_m \int_{V_m} H_J \delta_{ac} \rho \ H_I \dddot{u}_{Jc} dV^m + \\
\bigcup_m \int_{V_m} (H_{I,b}) \ E_{abcd} \ (H_{J,d} \dddot{u}_{Jc}) \ dV^m = \\
= \bigcup_m \int_{V_m} f_a^B \ H_I \ dV^m + \bigcup_m \int_{S_m} f_a^S \ H_I \ dS^m + \\
+ \bigcup_m \int_{V_m} (E_{abcd} \epsilon_{cd}^0 H_{I,b}) \ dV^m - \bigcup_m \int_{V_m} (\sigma_{ab}^0) \ H_{I,b} \ dV^m
\]

(102.26)

One should also observe that in the first line of equation (102.26) generalized nodal accelerations \( \dddot{u}_{Jc} \) and generalized nodal displacements \( \dddot{u}_{Jc} \) are unknowns that are not subjected to integration so they can be factored out of the integral:

\[
\bigcup_m \int_{V_m} H_J \delta_{ac} \rho \ H_I \ dV^m \dddot{u}_{Jc} + \\
\bigcup_m \int_{V_m} H_{I,b} \ E_{abcd} \ H_{J,d} \ dV^m \dddot{u}_{Jc} = \\
= \bigcup_m \int_{V_m} f_a^B \ H_I \ dV^m + \bigcup_m \int_{S_m} f_a^S \ H_I \ dS^m + \\
+ \bigcup_m \int_{V_m} (E_{abcd} \epsilon_{cd}^0 H_{I,b}) \ dV^m - \bigcup_m \int_{V_m} (\sigma_{ab}^0) \ H_{I,b} \ dV^m
\]

(102.27)

We can now define several tensors from equation (102.27):

\[
^{(m)}M_{IacJ} = \int_{V_m} H_J \delta_{ac} \rho \ H_I \ dV^m (102.28)
\]

\[
^{(m)}K_{IacJ} = \int_{V_m} H_{I,b} \ E_{abcd} \ H_{J,d} \ dV^m (102.29)
\]
\[(m)F^B_{Ia} = \int_{V} f^B_a H_I dV^m \quad (102.30)\]

\[(m)F^S_{Ia} = \int_{S} f^S_a H_I dS^m \quad (102.31)\]

\[(m)F^{0\epsilon}_{mnIa} = \int_{V} E_{abcd} \epsilon^0_{cd} H_{I,b} dV^m \quad (102.32)\]

\[(m)F^{0\sigma}_{mnIa} = \int_{V} \sigma^0_{ab} H_{I,b} dV^m \quad (102.33)\]

where \((m)M_{IacJ}\) is the element mass tensor, \((m)K_{IacJ}\) is the element stiffness tensor, \((m)F^B_{Ia}\) is the tensor of element body forces, \((m)F^S_{Ia}\) is the tensor of element surface forces, \((m)F^{0\epsilon}_{mnIa}\) is the tensor of element initial strain effects, \((m)F^{0\sigma}_{mnIa}\) is the tensor of element initial stress effects. Now equation \((102.27)\) becomes:

\[
\bigcup (m)M_{IacJ} \dddot{u}_{Jc} + \bigcup (m)K_{IacJ} \ddot{u}_{Jc} = \bigcup m(m)F^B_{Ia} + \bigcup m(m)F^S_{Ia} + \bigcup m(m)F^{0\epsilon}_{mnIa} - \bigcup m(m)F^{0\sigma}_{mnIa} \quad (102.34)
\]

By summing\(^{13}\) all the relevant tensors, a well known equation is obtained:

\[
M_{AacB} \dddot{u}_{Bc} + K_{AacB} \ddot{u}_{Bc} = F_Aa \quad (102.35)
\]

\[A, B = 1, 2, \ldots, \# \text{ of nodes}\]

\[a, c = 1, \ldots, \# \text{ of dimensions (1, 2 or 3)}\]

\(^{13}\)Summation of the element volume integrals expresses the direct addition of the element tensors to obtain global, system tensors. This method of direct addition is usually referred to as the direct stiffness method.
where:

\[
M_{AacB} = \bigcup_m^{(m)} M_{IacJ}; \quad K_{AacB} = \bigcup_m^{(m)} K_{IacJ}
\] (102.36)

are the system mass and stiffness tensors, respectively, \(\ddot{u}_{Bc}\) is the tensor of unknown nodal accelerations, and \(\ddot{u}_{Bc}\) is the tensor of unknown generalized nodal displacements, while the load tensor is given as:

\[
F_{Aa} = \bigcup_m^{(m)} F_{Ia}^B + \bigcup_m^{(m)} F_{Ia}^S + \bigcup_m^{(m)} F_{Ia}^{\sigma_0} - \bigcup_m^{(m)} F_{Ia}^{\sigma_0}
\] (102.37)

After assembling the system of equations in (102.36), it is relatively easy to solve for the unknown displacements \(\ddot{u}_{Lc}\) either for static or fully dynamic case. It is also very important to note that in all previous equations, omissions of inertial force term (all terms with \(\rho\)) will yield static equilibrium equations. Description of solutions procedures for static linear and nonlinear problems are described in some detail in chapter 107. In addition to that, solution procedures for dynamic, linear and nonlinear problems are described in some detail in chapter 108.

A note on the final form of the tensors used is in order. In order to use readily available system of equation solvers equation (102.36) will be rewritten in the following form:

\[
M_{PQ} \ddot{u}_P + K_{PQ} \ddot{u}_P = F_Q \quad P, Q = 1, 2, \ldots, (\# of DOFs)N
\] (102.38)

where \(M_{PQ}\) is system mass matrix, \(K_{PQ}\) is system stiffness matrix and \(F_Q\) is the loading vector. Matrix form of equation 102.36, presented as equation 102.38 is obtained flattening the system mass tensor \(M_{AacB}\), system stiffness tensor \(K_{AacB}\), unknown acceleration tensor \(\ddot{u}_{Bc}\), unknown displacement tensor \(\ddot{u}_{Bc}\) and the system loading tensor \(F_{Aa}\). Flattening from the fourth order mass/stiffness tensors to two-dimensional mass/stiffness matrix is done by simply performing appropriate (re-)numbering of nodal DOFs in each dimension. A similar approach is used for unknown accelerations/displacements and loadings.

### 102.3.1 Static Analysis: Internal and External Loads.

Internal and external loading tensors is defined as:
\[(f_{Ia})_{int} = \bigcup_{(m)}^{(m)} K_{IacJ} \bar{u}_{Jc} = \bigcup_{m} \int_{V_m} \sigma_{ab} H_{I,b} \, dV^m \]  
(102.39)

\[(f_{Ia})_{ext} = \bigcup_{m}^{(m)} F_{Ia}^B + \bigcup_{m}^{(m)} F_{Ia}^S + \bigcup_{m}^{(m)} F_{Ia}^{\sigma_{0mn}} - \bigcup_{m}^{(m)} F_{Ia}^{\sigma_{0mn}} \]  
(102.40)

where \((f_{Ia})_{int}\) is the internal force tensor and \((f_{Ia})_{ext}\) is the external force tensor. Equilibrium is obtained when residual:

\[r_{Ia}(\bar{u}_{Jc}, \lambda) = (f_{Ia}(\bar{u}_{Jc}))_{int} - \lambda (f_{Ia})_{ext} \]  
(102.41)

is equal to zero, \(r(u, \lambda) = 0\). The same equation in flattened form yields:

\[r(u, \lambda) = f_{int}(u) - \lambda f_{ext} = 0 \]  
(102.42)
102.4 Isoparametric Solid Finite Elements

102.4.1 8 Node Brick

Table 102.1: Values of $r_1$, $r_2$, and $r_3$ at each of the eight nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>4</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>5</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>8</td>
<td>+1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Shape function of the nodes which $i$ indicates the node number:

$$N_i^{(e)} = \frac{1}{8}(1 + r_1(r_1)_i)(1 + r_2(r_2)_i)(1 + r_3(r_3)_i)$$  \hspace{1cm} (102.43)
102.4.2 Collapsed 8 Node Brick

It is sometimes required to mesh finite element models using collapsed brick elements. Collapsed brick elements are finite elements that do not feature all 8 nodes, rather some nodes are merged. This is done to help generate meshes for complicated geometries where it is impossible to rely on solid bricks with eight (8) nodes only. For example, SASSI2000 (System for Analysis of Soil-Structure Interaction) program (Ostadan, 2007) uses such elements. For example, solid elements with 7, 6, and 5 nodes are used extensively and are created by collapsing/combining nodes of 8 node brick into solid elements with 7, 6 or 5 nodes. There are three types of collapsed SASSI 8 node brick element, as shown in Figure 102.4.

![Collapsed 8 Node Brick](image)

Figure 102.4: Three types of collapsed brick elements in SASSI (Ostadan, 2007).

Verification tests for collapsed brick finite elements are provided in verification section ?? on page 1482.
102.4.3 20 Node Brick

![20 Node Brick Diagram](image)

Figure 102.5: 20 node brick element

Table 102.2: Values of $r_1$, $r_2$, and $r_3$ at each of the $9^{th}$ to $20^{th}$ nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>10</td>
<td>-1</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>12</td>
<td>+1</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>14</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>16</td>
<td>+1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>17</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>-1</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>
Shape function of the 8 corner nodes (1 to 8) which $i$ indicates the node number:

\[ N_i^{(e)} = \frac{1}{8}(1 + r_1(r_1)_i)(1 + r_2(r_2)_i)(1 + r_3(r_3)_i)(r_1(r_1)_i + r_2(r_2)_i + r_3(r_3)_i - 2) \]  \hfill (102.44)

Shape function of the node numbers 9, 11, 13, and 15 which $i$ indicates the node number:

\[ N_i^{(e)} = \frac{1}{4}(1 - r_1^2)(1 + r_2(r_2)_i)(1 + r_3(r_3)_i) \]  \hfill (102.45)

Shape function of the node numbers 10, 12, 14, and 16 which $i$ indicates the node number:

\[ N_i^{(e)} = \frac{1}{4}(1 - r_2^2)(1 + r_1(r_1)_i)(1 + r_3(r_3)_i) \]  \hfill (102.46)

Shape function of the node numbers 17, 18, 19, and 20 which $i$ indicates the node number:

\[ N_i^{(e)} = \frac{1}{4}(1 - r_3^2)(1 + r_1(r_1)_i)(1 + r_2(r_2)_i) \]  \hfill (102.47)
102.4.4 27 Node Brick

Shape function of the 8 corner nodes (1 to 8) which \( i \) indicates the node number:

\[
N_i^{(e)} = \frac{1}{8}(1 + r_1(r_1)_i)(1 + r_2(r_2)_i)(1 + r_3(r_3)_i)(r_1(r_1)_i)(r_2(r_2)_i)(r_3(r_3)_i) \quad \text{(102.48)}
\]

Shape function of the node numbers 9, 11, 13, and 15 which \( i \) indicates the node number:

\[
N_i^{(e)} = \frac{1}{4}(1 - r_1^2)(1 + r_2(r_2)_i)(1 + r_3(r_3)_i)(r_2(r_2)_i)(r_3(r_3)_i) \quad \text{(102.49)}
\]
Shape function of the node numbers 10, 12, 14, and 16 which $i$ indicates the node number:

$$N^{(e)}_i = \frac{1}{4}(1 + r_1(r_1)_i)(1 - r_2^2)(1 + r_3(r_3)_i)(r_1(r_1)_i)(r_3(r_3)_i)$$ (102.50)

Shape function of the node numbers 17, 18, 19, and 20 which $i$ indicates the node number:

$$N^{(e)}_i = \frac{1}{4}(1 + r_1(r_1)_i)(1 + r_2(r_2)_i)(1 - r_3^2)(r_1(r_1)_i)(r_2(r_2)_i)$$ (102.51)

Shape function of the node number 21:

$$N^{(e)}_{21} = (1 - r_1^2)(1 - r_2^2)(1 - r_3^2)$$ (102.52)

Shape function of the node numbers 22 and 24 which $i$ indicates the node number:

$$N^{(e)}_i = \frac{1}{2}(1 - r_1^2)(1 + r_2(r_2)_i)(1 - r_3^2)(r_2(r_2)_i)$$ (102.53)

Shape function of the node numbers 23 and 25 which $i$ indicates the node number:

$$N^{(e)}_i = \frac{1}{2}(1 + r_1(r_1)_i)(1 - r_2^2)(1 - r_3^2)(r_1(r_1)_i)$$ (102.54)

Shape function of the node numbers 26 and 27 which $i$ indicates the node number:

$$N^{(e)}_i = \frac{1}{2}(1 - r_1^2)(1 - r_2^2)(1 + r_3(r_3)_i)(r_3(r_3)_i)$$ (102.55)

### 102.4.5 Isoparametric 8 – 20 Node Finite Element

The basic procedure in the isoparametric\(^{14}\) finite element formulation is to express the element coordinates and element displacements in the form of interpolations using the local three dimensional\(^{15}\) coordinate system of the element. Considering the general 3D element, the coordinate interpolations, using indicial notation\(^{16}\) are:

\(^{14}\)name isoparametric comes from the fact that both displacements and coordinates are defined in terms of nodal values. Superparametric and subparametric finite elements exists also.

\(^{15}\)in the case of element presented here, that is isoparametric 8 – 20 node finite element.

\(^{16}\)Einstein’s summation rule is implied unless stated differently, all lower case indices ($i, j, p, q, m, n, o, r, s, t, \ldots$) can have values of 1, 2, 3, and values for capital letter indices will be specified where need be.
\[ x_i = H_A(r_k) \bar{x}_{Ai} \]  

(102.56)

where \( A = 1, 2, \ldots, n \) and \( n \) is the total number of nodes associated with that specific element,  \( \bar{x}_{Ai} \) is the \( i \)-th coordinate of node \( A, i = 1, 2, 3, \) \( k = 1, 2, 3 \) and \( H_A \) are the interpolation functions defined in local coordinate system of the element, with variables \( r_1, r_2 \) and \( r_3 \) varying from \(-1\) to \(+1\).

The interpolation functions \( H_A \) for the isoparametric 8–20 node are the so called serendipity interpolation functions mainly because they were derived by inspection. For the finite element with nodes numbered as in Figure (102.7) they are given\(^{17}\) in the following set of formulae:

\[
\begin{align*}
H_{20} &= \frac{isp(20)}{4} (1 + r_1) (1 - r_2) (1 - r_3^2) \\
H_{18} &= \frac{isp(18)}{4} (1 - r_1) (1 + r_2) (1 - r_3^2) \\
H_{16} &= \frac{isp(16)}{4} (1 + r_1) (1 - r_2^2) (1 - r_3) \\
H_{14} &= \frac{isp(14)}{4} (1 - r_1) (1 - r_2^2) (1 - r_3) \\
H_{12} &= \frac{isp(12)}{4} (1 + r_1) (1 - r_2^2) (1 + r_3) \\
H_{10} &= \frac{isp(10)}{4} (1 - r_1) (1 - r_2^2) (1 + r_3) \\
H_{19} &= \frac{isp(19)}{4} (1 - r_1) (1 - r_2) (1 - r_3^2) \\
H_{17} &= \frac{isp(17)}{4} (1 + r_1) (1 + r_2) (1 - r_3^2) \\
H_{15} &= \frac{isp(15)}{4} (1 - r_1^2) (1 - r_2) (1 - r_3) \\
H_{13} &= \frac{isp(13)}{4} (1 - r_1^2) (1 + r_2) (1 - r_3) \\
H_{11} &= \frac{isp(11)}{4} (1 - r_1^2) (1 - r_2) (1 + r_3) \\
H_{9} &= \frac{isp(9)}{4} (1 - r_1^2) (1 + r_2) (1 + r_3)
\end{align*}
\]

\(^{17}\) for more details see Bathe (1982).
\[ H_8 = \frac{(1 + r_1) (1 - r_2) (1 - r_3)}{8} + \frac{-H_{15} - H_{16} - H_{20}}{2} \]
\[ H_7 = \frac{(1 - r_1) (1 - r_2) (1 - r_3)}{8} + \frac{-H_{14} - H_{15} - H_{19}}{2} \]
\[ H_6 = \frac{(1 - r_1) (1 + r_2) (1 - r_3)}{8} + \frac{-H_{13} - H_{14} - H_{18}}{2} \]
\[ H_5 = \frac{(1 + r_1) (1 + r_2) (1 - r_3)}{8} + \frac{-H_{13} - H_{16} - H_{17}}{2} \]
\[ H_4 = \frac{(1 + r_1) (1 - r_2) (1 + r_3)}{8} + \frac{-H_{11} - H_{12} - H_{20}}{2} \]
\[ H_3 = \frac{(1 - r_1) (1 - r_2) (1 + r_3)}{8} + \frac{-H_{10} - H_{11} - H_{19}}{2} \]
\[ H_2 = \frac{(1 - r_1) (1 + r_2) (1 + r_3)}{8} + \frac{-H_{10} - H_{18} - H_{9}}{2} \]
\[ H_1 = \frac{(1 + r_1) (1 + r_2) (1 + r_3)}{8} + \frac{-H_{12} - H_{17} - H_{9}}{2} \]

where \( r_1, r_2 \) and \( r_3 \) are the axes of natural, local, curvilinear coordinate system and \( isp(\text{nod}_\text{num}) \) is boolean function that returns +1 if node number \( \text{nod}_\text{num} \) is present and 0 if node number \( \text{nod}_\text{num} \) is not present.

To be able to evaluate various important element tensors, to calculate the strain–displacement transformation tensor is needed. The element strains are obtained in terms of derivatives of element displacements with respect to the local coordinate system. Because the element displacements are defined in the local coordinate system, there is a need to relate global \( x_1, x_2 \) and \( x_3 \) derivatives to the \( r_1, r_2 \) and \( r_3 \) derivatives. In order to obtain derivatives with respect to global coordinate system, i.e. \( \frac{\partial}{\partial x_a} \), use chain rule for differentiation in the following form:

\[
\frac{\partial}{\partial x_k} = \frac{\partial r_a}{\partial x_k} \frac{\partial}{\partial r_a} = J_{ak}^{-1} \frac{\partial}{\partial r_a}
\]

while the inverse relation is:

\[
\frac{\partial}{\partial r_k} = \frac{\partial x_a}{\partial r_k} \frac{\partial}{\partial x_a} = J_{ak} \frac{\partial}{\partial x_a}
\]

where \( J_{ak} \) is the Jacobian operator relating local coordinate derivatives to the global coordinate derivatives:

\(^{18}\)i.e. \((m)\mathbf{K}_{\text{int}}\), \((m)\mathbf{F}^{\overline{b}}_{\text{ia}}, (m)\mathbf{F}^\overline{a}_{\text{ia}}, (m)\mathbf{F}^a_{\text{ia}}\), \((m)\mathbf{F}^0\overline{m}_{\text{ia}}, (m)\mathbf{F}^0_{\text{ia}}\), that are defined in chapter (102.3).

\(^{19}\)from the equation \( \hat{\epsilon}_{ab} = \frac{1}{2} \((H_{1,b} \overline{u}_{ia}) + (H_{1,a} \overline{u}_{ib}) \)
\[
J_{ak} = \frac{\partial x_a}{\partial r_k} = \begin{bmatrix}
\frac{\partial x_1}{\partial r_1} & \frac{\partial x_2}{\partial r_1} & \frac{\partial x_3}{\partial r_1} \\
\frac{\partial x_1}{\partial r_2} & \frac{\partial x_2}{\partial r_2} & \frac{\partial x_3}{\partial r_2} \\
\frac{\partial x_1}{\partial r_3} & \frac{\partial x_2}{\partial r_3} & \frac{\partial x_3}{\partial r_3}
\end{bmatrix}
\]

The existence of equation (102.57) requires that the inverse of \( J_{ak} \) exists and that inverse exists provided that there is a one-to-one correspondence between the local and the global coordinates of element.

It should be pointed out that except for the very simple cases, volume and surface element tensor integrals are evaluated by means of numerical integration. Numerical integration rules is quite a broad subject and will not be covered here.

### 102.4.6 Isoparametric 8 - 27 Node Finite Element

![Figure 102.8: 8-27 variable node brick element](image)

---

\(^{20}\)unique.

\(^{21}\)as defined in chapter (102.3) by equations (102.29), (102.30), (102.31), (102.32) and (102.33).

\(^{22}\)Gauss–Legendre, Newton–Coates, Lobatto are among the most used integration rules.

\(^{23}\)nice explanation with examples is given in Bathe (1982).
\[
H_1 = \frac{(1 + r_1)(1 + r_2)(1 + r_3)}{8} - \frac{H_9 + H_{12} + H_{17}}{2} - \frac{H_{22} + H_{25} + H_{26}}{4} - \frac{H_{21}}{8}
\]

\[
H_2 = \frac{(1 - r_1)(1 + r_2)(1 + r_3)}{8} - \frac{H_9 + H_{10} + H_{18}}{2} - \frac{H_{22} + H_{23} + H_{26}}{4} - \frac{H_{21}}{8}
\]

\[
H_3 = \frac{(1 - r_1)(1 - r_2)(1 + r_3)}{8} - \frac{H_{10} + H_{11} + H_{19}}{2} - \frac{H_{23} + H_{24} + H_{26}}{4} - \frac{H_{21}}{8}
\]

\[
H_4 = \frac{(1 + r_1)(1 - r_2)(1 + r_3)}{8} - \frac{H_{11} + H_{12} + H_{20}}{2} - \frac{H_{24} + H_{25} + H_{26}}{4} - \frac{H_{21}}{8}
\]

\[
H_5 = \frac{(1 + r_1)(1 - r_2)(1 - r_3)}{8} - \frac{H_{13} + H_{16} + H_{17}}{2} - \frac{H_{22} + H_{25} + H_{27}}{4} - \frac{H_{21}}{8}
\]

\[
H_6 = \frac{(1 - r_1)(1 + r_2)(1 - r_3)}{8} - \frac{H_{13} + H_{14} + H_{18}}{2} - \frac{H_{22} + H_{23} + H_{27}}{4} - \frac{H_{21}}{8}
\]

\[
H_7 = \frac{(1 - r_1)(1 - r_2)(1 - r_3)}{8} - \frac{H_{14} + H_{15} + H_{19}}{2} - \frac{H_{23} + H_{24} + H_{27}}{4} - \frac{H_{21}}{8}
\]

\[
H_8 = \frac{(1 + r_1)(1 - r_2)(1 - r_3)}{8} - \frac{H_{15} + H_{16} + H_{20}}{2} - \frac{H_{24} + H_{25} + H_{27}}{4} - \frac{H_{21}}{8}
\]

\[
H_9 = \frac{1}{4}(1 - r_1^2)(1 + r_2)(1 + r_3) - \frac{H_{22} + H_{26}}{2} - \frac{H_{21}}{4}
\]

\[
H_{10} = \frac{1}{4}(1 - r_2^2)(1 - r_1)(1 + r_3) - \frac{H_{23} + H_{26}}{2} - \frac{H_{21}}{4}
\]

\[
H_{11} = \frac{1}{4}(1 - r_1^2)(1 - r_2)(1 + r_3) - \frac{H_{24} + H_{26}}{2} - \frac{H_{21}}{4}
\]

\[
H_{12} = \frac{1}{4}(1 - r_2^2)(1 + r_1)(1 + r_3) - \frac{H_{25} + H_{26}}{2} - \frac{H_{21}}{4}
\]

\[
H_{13} = \frac{1}{4}(1 - r_1^2)(1 + r_2)(1 - r_3) - \frac{H_{22} + H_{27}}{2} - \frac{H_{21}}{4}
\]

\[
H_{14} = \frac{1}{4}(1 - r_2^2)(1 - r_1)(1 - r_3) - \frac{H_{23} + H_{27}}{2} - \frac{H_{21}}{4}
\]

\[
H_{15} = \frac{1}{4}(1 - r_1^2)(1 - r_2)(1 - r_3) - \frac{H_{24} + H_{27}}{2} - \frac{H_{21}}{4}
\]

\[
H_{16} = \frac{1}{4}(1 - r_2^2)(1 + r_1)(1 - r_3) - \frac{H_{25} + H_{27}}{2} - \frac{H_{21}}{4}
\]

\[
H_{17} = \frac{1}{4}(1 - r_3^2)(1 + r_1)(1 + r_2) - \frac{H_{22} + H_{25}}{2} - \frac{H_{21}}{4}
\]

\[
H_{18} = \frac{1}{4}(1 - r_3^2)(1 - r_1)(1 + r_2) - \frac{H_{22} + H_{23}}{2} - \frac{H_{21}}{4}
\]

\[
H_{19} = \frac{1}{4}(1 - r_3^2)(1 - r_1)(1 - r_2) - \frac{H_{23} + H_{24}}{2} - \frac{H_{21}}{4}
\]

\[
H_{20} = \frac{1}{4}(1 - r_3^2)(1 + r_1)(1 - r_2) - \frac{H_{24} + H_{25}}{2} - \frac{H_{21}}{4}
\]

\[
H_{21} = (1 - r_1^2)(1 - r_2^2)(1 - r_3^2)
\]
To apply surface load on brick elements, equivalent nodal forces have to be applied instead of the surface load. The equivalent force of the $i$-th node $F_i$ is given by the following equation with shape function $H_i$ and load distribution function $f$.

$$F_i = \int_S f H_i \, ds$$  \hspace{1cm} (102.60)

Assuming that the load distribution is uniform

$$F_i = f \int_S H_i \, ds$$  \hspace{1cm} (102.61)

Furthermore, when the magnitude of the load per unit area is 1, and the size of the element is $1 \times 1 \times 1$, equivalent nodal forces are given as shown in Figure 102.9 for 8 node brick element, 20 brick element, and 27 nodes brick element.

Figure 102.9 shows cases of normal loads on vertical upper surface (with nodes: 1, 2, 3, 4 for 8 node brick; 1, 2, 3, 4, 9, 10, 11, and 12 for 20 node brick; and 1, 2, 3, 4, 9, 10, 11, 12 and 26 for the 27 node brick).

Nodal loads from uniform surface loads for 27 node brick are obtained as:

$$H_{22} = \frac{1}{2}(1 - r_1^3)(1 + r_2)(1 - r_3^3)r_2$$

$$H_{23} = -\frac{1}{2}(1 - r_1)(1 - r_2^3)(1 - r_3^3)r_1$$

$$H_{24} = -\frac{1}{2}(1 - r_1^3)(1 - r_2)(1 - r_3^3)r_2$$

$$H_{25} = \frac{1}{2}(1 + r_1)(1 - r_2^3)(1 - r_3^3)r_1$$

$$H_{26} = \frac{1}{2}(1 - r_1^3)(1 - r_2^3)(1 + r_3)r_3$$

$$H_{27} = -\frac{1}{2}(1 - r_1^3)(1 - r_2^3)(1 - r_3)r_3$$

### 102.4.7 Surface Loads for Solid Bricks
Figure 102.9: Nodal loads for brick elements: (a) \( F_1 = F_2 = F_3 = F_4 = +\frac{1}{4} \); (b) \( F_1 = F_2 = F_3 = F_4 = -\frac{1}{12} \); \( F_9 = F_{10} = F_{11} = F_{12} = +\frac{1}{3} \); (c) \( F_1 = F_2 = F_3 = F_4 = +\frac{1}{36} \); \( F_9 = F_{10} = F_{11} = F_{12} = +\frac{1}{9} \); \( F_{36} = \frac{4}{9} \).

- for nodes 1, 2, 3, and 4, \( N_i^{(e)} = \frac{1}{8}(1 + r_1(r_1)i)(1 + r_2(r_2)i)(1 + r_3(r_3)i)(r_1(r_1)i)(r_2(r_2)i)(r_3(r_3)i) \)

\[
\int_{-1}^{+1} H_i dS = \\
= \frac{1}{8}(1 + r_3(r_3)i)(r_3(r_3)i) \int_{-1}^{+1} \int_{-1}^{+1} (1 + r_1(r_1)i)(1 + r_2(r_2)i)(r_1(r_1)i)(r_2(r_2)i)(r_2(r_2)i)dr_1 dr_2 \\
= \frac{1}{8}(1 + r_3(r_3)i)(r_3(r_3)i)((r_1)i)^2((r_2)i)^2(\frac{2}{3})^2 \\
= \frac{1}{18}(1 + r_3(r_3)i)(r_3(r_3)i)((r_1)i)^2((r_2)i)^2
\]

- for nodes 9, 10, 11 and 12, \( N_i^{(e)} = \frac{1}{8}(1 - r_1^2)(1 + r_2(r_2)i)(1 + r_3(r_3)i)(r_2(r_2)i)(r_3(r_3)i) \)

\[
\int_{-1}^{+1} H_i dS = \\
= \frac{1}{4}(1 + r_3(r_3)i)(r_3(r_3)i) \int_{-1}^{+1} (1 - r_1^2)(1 + r_2(r_2)i)(r_2(r_2)i)dr_1 dr_2 \\
= \frac{1}{8}(1 + r_3(r_3)i)(r_3(r_3)i)((r_2)i)^2(\frac{4}{3})(\frac{2}{3}) \\
= \frac{2}{9}(1 + r_3(r_3)i)(r_3(r_3)i)((r_2)i)^2
\]
• for nodes 26 \( N_{i}^{(e)} = \frac{1}{2}(1 - r_{1}^{2})(1 - r_{2}^{2})(1 + r_{3}(r_{3})_{i})(r_{3}(r_{3})_{i}) \)

\[ \int_{-1}^{+1} H_{i} dS = \]

\[ = \frac{1}{4}(1 + r_{3}(r_{3})_{i})(r_{3}(r_{3})_{i}) \int_{-1}^{+1} \int_{-1}^{+1} (1 - r_{1}^{2})(1 - r_{2}^{2}) dr_{1} dr_{2} \]

\[ = \frac{1}{8}(1 + r_{3}(r_{3})_{i})(r_{3}(r_{3})_{i}) \left( \frac{4}{3} \right) \left( \frac{4}{3} \right) \]

\[ = \frac{8}{9}(1 + r_{3}(r_{3})_{i})(r_{3}(r_{3})_{i}) \quad (102.64) \]

102.5 Numerical Integration for Solid Brick Elements

Gauss integration rule, see Bathe (1996), section 5.5.3. While using the regular Newton-Coates integration formula, one uses \((n + 1)\) equally spaced points to integrate exactly polynomial of order \(n\). On the other hand, while using the Gauss integration formula, one uses \(n\) unequally spaced points to integrate exactly polynomial of order \((2n - 1)\).

102.6 Two Node, 3D Truss Finite Element

Bathe and Wilson (1976); Bathe (1982)

102.7 3D Beam-Column Finite Element, 12 Degrees of Freedom

Bathe and Wilson (1976); Bathe (1982); Przemieniecki (1985)

Stiffness Matrix: Equation 102.65
Mass Matrix: Equation 102.66
\[
[K] = \begin{bmatrix}
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
0 & \frac{6EA}{L^2} & 0 & 0 & 0 & \frac{6EA}{L^2} & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
\[
[M] = \begin{bmatrix}
\rho AL & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3}
\end{bmatrix}
\]
102.8 3D Beam-Column Finite Element, 9 Degrees of Freedom

Przemieniecki (1985)

Condensation Formulation: Equations 102.67 to 102.73
Rearranged 12dof Stiffness Matrix: Equation 102.74
K_{rr} part of stiffness matrix: Equation 102.75
K_{rc} part of stiffness matrix: Equation 102.76
K_{cr} part of stiffness matrix: Equation 102.77
K_{cc} part of stiffness matrix: Equation 102.78
Stiffness Matrix: Equation 102.79
T Matrix: Equation 102.80
Rearranged Mass Matrix: Equation 102.81
Mass Matrix: Equation 102.82

\[
\begin{bmatrix}
  k_{rr} & k_{rc} \\
  k_{cr} & k_{cc}
\end{bmatrix}
\begin{bmatrix}
  d_r \\
  d_c
\end{bmatrix}
= \begin{bmatrix}
  r_r \\
  r_c
\end{bmatrix}
\]

(102.67)

\[
\begin{bmatrix}
  k_{rr}
\end{bmatrix}
- \begin{bmatrix}
  K_{cc}
\end{bmatrix}^{-1}
\begin{bmatrix}
  K_{rc}
\end{bmatrix}
\begin{bmatrix}
  d_r
\end{bmatrix}
= \begin{bmatrix}
  r_r
\end{bmatrix}
- \begin{bmatrix}
  K_{rc}
\end{bmatrix}^{-1}
\begin{bmatrix}
  r_c
\end{bmatrix}
\]

(102.68)

\[
[k_{\text{condensed}}] = \begin{bmatrix}
  k_{rr}
\end{bmatrix}
- \begin{bmatrix}
  K_{rc}
\end{bmatrix}^{-1}
\begin{bmatrix}
  K_{cr}
\end{bmatrix}
\]

(102.69)

\[
r_{\text{condensed}} = \begin{bmatrix}
  r_r
\end{bmatrix}
- \begin{bmatrix}
  K_{rc}
\end{bmatrix}^{-1}
\begin{bmatrix}
  r_c
\end{bmatrix}
\]

(102.70)

\[
[T] = \begin{bmatrix}
  I \\
  \begin{bmatrix}
  K_{cc}
\end{bmatrix}^{-1}
\begin{bmatrix}
  K_{cr}
\end{bmatrix}
\end{bmatrix}
\]

(102.71)

\[
[K_{\text{condensed}}] = [T]^T[K][T]
\]

(102.72)

\[
[M_{\text{condensed}}] = [T]^T[M][T]
\]

(102.73)

K_{\text{condensed}} should give the same results using either method.
\[
\begin{bmatrix}
EA \frac{L}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{E_t}{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{E_t}{L} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{E_t}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{E_t}{L} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{E_t}{L} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{E_t}{L} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_t}{L} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{E_t}{L}
\end{bmatrix}
\]

\[K_{\text{rearranged}} = (102.74)\]
\[
[K_{rr}] = \begin{bmatrix}
\frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12EI_z}{L^3} & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & 0 & 0 & \frac{6EI_y}{L^2} \\
0 & 0 & \frac{12EI_y}{L^3} & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\
-\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12EI_z}{L^3} & 0 & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\
0 & 0 & -\frac{12EI_y}{L^3} & 0 & 0 & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & G\frac{J_z}{L} & 0 & 0 \\
0 & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\
0 & \frac{6EI_z}{L^2} & 0 & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L}
\end{bmatrix}
\tag{102.75}
\]

\[
[K_{rc}] = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{GJ_z}{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2EI_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2EI_z & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\tag{102.76}
\]

\[
[K_{cr}] = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{GJ_z}{L} & 0 \\
0 & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & \frac{6EI_z}{L^2} & 0 & 2EI_y & 0 \\
0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{2EI_z}{L} & 0 & 0 
\end{bmatrix}
\tag{102.77}
\]

\[
[K_{cc}] = \begin{bmatrix}
\frac{GJ_z}{L} & 0 & 0 & 0 \\
0 & \frac{4EI_z}{L} & 0 & 0 \\
0 & 0 & \frac{4EI_z}{L}
\end{bmatrix}
\tag{102.78}
\]
\[
[K_{\text{condensed}}] = \begin{bmatrix}
\frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{3EI_z}{L^3} & 0 & 0 & -\frac{3EI_z}{L^3} & 0 & 0 & 0 & \frac{3EI_z}{L^3} \\
0 & 0 & 3EI_y/L^3 & 0 & 0 & 3EI_y/L^3 & 0 & 0 & 3EI_y/L^3 \\
-\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{3EI_z}{L^3} & 0 & 0 & 3EI_z/L^3 & 0 & 0 & 0 & -\frac{3EI_z}{L^3} \\
0 & 0 & -\frac{3EI_y}{L^3} & 0 & 0 & 3EI_y/L^3 & 0 & 0 & 3EI_y/L^3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{3EI_y}{L^2} & 0 & 0 & 3EI_y/L^2 & 0 & 0 & 3EI_y/L \\
0 & \frac{3EI_z}{L^3} & 0 & 0 & -\frac{3EI_z}{L^3} & 0 & 0 & 0 & \frac{3EI_z}{L} \\
\end{bmatrix}
\] (102.79)

\[
[T] = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & \frac{3}{2L} & 0 & 0 & -\frac{3}{2L} & 0 & -\frac{1}{2} & 0 \\
0 & -\frac{3}{2L} & 0 & 0 & \frac{3}{2L} & 0 & 0 & 0 & -\frac{1}{2} \\
\end{bmatrix}
\] (102.80)
\[ [M_{\text{rearranged}}] = \rho AL \]

\[
\begin{bmatrix}
1 & 1 & 3 & 6 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 11 & -\frac{J_y}{210} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{J_y}{210} & 0 & 11 & -\frac{J_y}{210} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & -\frac{J_y}{210} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{J_y}{210} & 11 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{J_y}{210} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
\[ [M_{\text{condensed}}] = \rho AL \]

\[
\begin{bmatrix}
\frac{1}{3} & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{33}{140} + \frac{6I_x}{5AL^2} & 0 & 0 & \frac{39}{280} - \frac{6I_z}{5AL^2} & 0 & 0 & 0 & -\frac{11L}{280} + \frac{I_x}{5AL} \\
0 & 0 & \frac{33}{140} + \frac{6I_y}{5AL^2} & 0 & 0 & \frac{39}{280} - \frac{6I_y}{5AL^2} & 0 & \frac{11L}{280} - \frac{I_y}{5AL} & 0 \\
\frac{1}{6} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{39}{280} - \frac{6I_x}{5AL^2} & 0 & 0 & \frac{17}{35} + \frac{6I_z}{5AL^2} & 0 & 0 & 0 & -\frac{3L}{35} - \frac{I_x}{5AL} \\
0 & 0 & \frac{39}{280} - \frac{6I_y}{5AL^2} & 0 & 0 & \frac{17}{35} + \frac{6I_y}{5AL^2} & 0 & \frac{3L}{35} + \frac{I_y}{5AL} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{J_y}{A} & 0 & 0 \\
0 & 0 & \frac{11L}{280} - \frac{I_y}{5AL} & 0 & 0 & \frac{3L}{35} + \frac{I_y}{5AL} & 0 & \frac{2L^2}{105} + \frac{I_y}{5A} & 0 \\
0 & -\frac{11L}{280} + \frac{I_x}{5AL} & 0 & 0 & -\frac{3L}{35} - \frac{I_x}{5AL} & 0 & 0 & 0 & \frac{2L^2}{105} + \frac{I_x}{5A}
\end{bmatrix}
\]
102.9 Shear Beam Finite Element

102.10 Quadrilateral Shell Finite Element with 6DOFs per Node

Based on works by Bergan and Felippa (1985); Alvin et al. (1992); Felippa and Militello (1992); Felippa and Alexander (1992); Militello and Felippa (1991). The stiffness matrix for this element is obtained by averaging two quad shells made up of two ANDES triangular shells (with an alternating orientation of diagonals, Stošić (1984-2013))
102.11 Seismic Isolator and Dissipator Finite Elements

Base isolation systems are used to change dynamic characteristics of seismic motions that excite structure and also to dissipate seismic energy before it excites structure. Therefore there are two main types of devices:

- **Base Isolators** (Kelly, 1991a,b; Toopchi-Nezhad et al., 2008; Huang et al., 2010; Vassiliou et al., 2013) are usually made of low damping (energy dissipation) elastomers and are primarily meant to change (reduce) frequencies of input motions. They are not designed nor modeled as energy dissipators.

- **Base Dissipators** Kelly and Hodder (1982); Fadi and Constantinou (2010); Kumar et al. (2014) are developed to dissipate seismic energy before it excites the structure. There are two main types of such dissipators:
  - Elastomers made of high dissipation rubber, and
  - Frictional pendulum dissipators

Both isolators and dissipators are usually developed to work in two horizontal dimensions, while motions in vertical direction are not isolated or dissipated. This can create potential problems and need to be carefully modeled.

Modeling of base isolation and dissipation system is done using two-node finite elements of relatively short length.

102.11.1 Base Isolation Systems

Base isolation systems are modeled using linear or nonlinear elastic elements. Stiffness is provided from either tests on a full-sized base isolators, or from material characterization of rubber (and steel plates if used in a sandwich isolator construction). Depending on rubber used, a number of models can be used to develop stiffness of the device Ogden (1984); Simo and Miehe (1992); Simo and Pister (1984).

Particularly important is to properly account for vertical stiffness as vertical motions can be amplified depending on characteristics of seismic motions, structure and stiffness of the isolators Hijikata et al. (2012); Araki et al. (2009). It is also important to note that assumption of small deformation is used in most cases. In other words, the stability of the isolator, for example, overturning or rolling is not modeled. It is assumed that elastic stiffness will not suddenly change if the isolator becomes unstable (rolls or overturns).
102.11.2 Base Dissipator Systems

Base dissipator systems are modeled using inelastic (nonlinear) two node elements. There are three basic types of dissipator models used:

- High damping rubber dissipators
- Rubber dissipators with lead core
- Frictional pendulum (double or triple) dissipators

Each one is calibrated using tests done on a full dissipator. It is important to be able to take into account the influence of (an increase in) temperature on resulting behavior. Energy dissipation results in heating of devices, and an increase in temperature influences material properties of dissipators.

102.11.3 Two Node, 3D, Rubber Isolator Finite Element

Kelly (1991a,b)

Behavior of rubber (Ogden, 1984; Simo and Miehe, 1992; Simo and Pister, 1984)

102.11.4 Two Node, 3D, Frictional Pendulum Finite Element

102.12 Fully Coupled, Porous Solid – Pore Fluid Finite Elements

102.12.1 u-p-U Formulation

102.12.1.1 Background

This section follows developments by Zienkiewicz and Shiomi (1984).

The relationship between effective stress, total stress and pore pressure is given as:

\[ \sigma''_{ij} = \sigma_{ij} - \alpha \delta_{ij} p \]  \hspace{1cm} (102.83)

where \( \sigma''_{ij} \) is effective stress tensor, \( \sigma_{ij} \) is total stress tensor, \( \delta_{ij} \) is Kronecker delta. \( \delta_{ij} = 1 \), when \( i=j \), and \( \delta_{ij} = 0 \), when \( i \neq j \). It is assumed that tensile components of effective and total stress are positive, and the pore fluid pressure \( p \) is also positive in tension, hence for compressions (usual case) pore fluid pressure is negative \( (p < 0) \) (Zienkiewicz et al., 1999a). For isotropic materials, \( \alpha = 1 - K_T/K_S \). \( K_T \) is the total bulk modulus of the solid matrix, \( K_S \) is the bulk modulus of the solid particle. For most of the soil mechanics problems, as the bulk modulus \( K_S \) of the solid particles is much larger than that of the whole material, \( \alpha \approx 1 \) can be assumed. Equation (102.83) becomes

\[ \sigma''_{ij} = \sigma_{ij} - \delta_{ij} p \]  \hspace{1cm} (102.84)
In the next sections, a detailed derivation of formulation and numerical implementation for a fully coupled (pore fluid and porous soil) solid mechanics problem is given. Derivations are based in part on earlier work by Zienkiewicz et al. (1999a).

102.12.1.2 Governing Equations of Porous Media

The following notation is used:

• $\sigma_{ij}$, the total Cauchy stress in the mixture,
• $u_i$, the displacement of the solid skeleton,
• $w_i$, the displacement of the fluid phase relative to the skeleton of solid,
• $p$, the pore water pressure,
• $\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i})$, the strain increment of the solid phase,
• $\omega_{ij} = \frac{1}{2}(u_{i,j} - u_{j,i})$, the rotation increment of the solid phase,
• $\rho, \rho_s, \rho_f$, the densities of the mixture, solid phase and water respectively,
• $n$ the porosity,
• $\theta = -\dot{w}_{i,i}$, the rate of change of volume of water per unit total volume of mixture.

The Equilibrium Equation of the Mixture. The overall equilibrium or momentum balance equation for the soil-fluid mixture is written as

$$\sigma_{ij,j} - \rho \ddot{u}_i - \rho_f \left[ \ddot{w}_i + \dot{w}_j \dot{w}_{i,j} \right] + \rho b_i = 0 \quad (102.85)$$

here $\ddot{u}_i$ is the acceleration of the solid part, $b_i$ is the body force per unit mass, $\ddot{w}_i + \dot{w}_j \dot{w}_{i,j}$ is the fluid acceleration relative to the solid part, $\dot{w}_i$ is local acceleration, $\dot{w}_j \dot{w}_{i,j}$ is convective acceleration.

The underlined terms in the above equation represent the fluid acceleration relative to the solid and convective terms of this acceleration. Generally, this acceleration is so small that it is frequently omitted. For static problems, equation 102.85 only consists of the first and last terms.

For fully saturated porous media (no air inside), from definition
\[ \rho = \frac{M_t}{V_t} = \frac{M_s + M_f}{V_t} = \frac{V_s \rho_s + V_f \rho_f}{V_t} = \frac{V_f \rho_f}{V_t} + \frac{V_t - V_f}{V_t} \rho_s = n \rho_f + (1 - n) \rho_s \]

where \( M_t, M_s \) and \( M_f \) are the mass of total, solid part and fluid part respectively. \( V_t, V_s \) and \( V_f \) are the volume of total, solid part and fluid part respectively.

**The Equilibrium Equation of the Fluid.** For the pore fluid, the equation of momentum balance is written as

\[ -p_i - R_i - \rho_f \ddot{u}_i - \rho_f [\ddot{w}_i + \dot{w}_j \dddot{w}_{i,j}] / n + \rho_f b_i = 0 \]

(102.87)

where \( R \) is the viscous drag forces. It is noted that the underlined terms in equation 102.87 represent the convective fluid acceleration again and are generally small. Also note that the permeability \( k \) is used with dimensions of \([\text{length}]^2[\text{time}]/[\text{mass}]\), which is different from the usual soil mechanics convention, where the permeability has the dimension of velocity, i.e., \([\text{length}]/[\text{time}]\). Their values are related by \( k = K/\rho_f g \), where \( g \) is the gravitational acceleration at which the permeability is measured. Assuming the Darcy seepage law: \( n \dot{w} = K \dot{i} \), here \( i \) is the head gradient. Seepage force is then \( R = \rho_f g i \). \( R \) can be written as

\[ R_i = k_{ij}^{-1} \ddot{w}_j \quad \text{or} \quad R_i = k^{-1} \ddot{w}_i \]

(102.88)

where \( k_{ij} \) or \( k \) are Darcy permeability coefficients for anisotropic and isotropic conditions respectively.

**Flow Conservation Equation.** The final equation is supplied by the mass conservation of the fluid flow

\[ \dot{w}_{i,i} + \alpha \dot{\varepsilon}_{ii} + \frac{\dot{p}}{Q} + n \frac{\dot{\rho}_f}{\rho_f} + \dot{s}_0 = 0 \]

(102.89)

The first term of equation(102.89) is the flow divergence of a unit volume of mixture. The second term is the volume change of the mixture. In the third term, \( Q \) is relative to the compressibility of the solid
and fluid. The underlined terms represent change of density and rate of volume expansion of the solid in case of thermal changes. They are generally negligible.

\[
\frac{1}{Q} \approx \frac{n}{K_f} + \frac{\alpha - n}{K_s} = \frac{1 - n}{K_s}
\]  

(102.90)

where \(K_s\) and \(K_f\) are the bulk moduli of the solid and fluid phases respectively. Note that the bulk modulus of the solid phase \(K_s\) is the actual bulk modulus of the solid particle.

Obtained are the total mixture equilibrium equation (102.85), fluid equilibrium equation (102.87) and the flow conservation equation (102.89) for saturated soil. By omitting the convective acceleration (the underline terms in (102.85) and (102.87)), density variation and the volume expansion due to the thermal change (the underline terms in (102.89)), the equations of coupled system can be further simplified, as summarized below

\[
\sigma_{ij,j} - \rho \ddot{u}_i - \rho_f \ddot{w}_i + \rho b_i = 0
\]

(102.91)

\[
-p_i - R_i - \rho_f \ddot{u}_i - \frac{\rho_f \ddot{w}_i}{n} + \rho_f b_i = 0
\]

(102.92)

\[
\dot{w}_{ii} + \alpha \dot{\varepsilon}_{ii} + \frac{\dot{p}}{Q} = 0
\]

(102.93)

**Bulk Modulus of Fluid** (see Verruijt (2012) page 97... compressibility of water and with air bubbles)...

### 102.12.1.3 Modified Governing Equations.

**Solid Part Equilibrium Equation.** A new variable \(U_i\) is introduced in place of the relative pseudo-displacement \(w_i\)

\[
U_i = u_i + U_i^R = u_i + \frac{w_i}{n}
\]

(102.94)

Figure 102.10: Fluid mechanics of Darcy’s flow \((w_i)\) versus real flow \((U_i = w_i/n)\).
Change of variables is finalized by insertion of equation 102.94 into equations 102.91 and 102.92, and then by subtraction of term \([n \times \text{equation } 102.92]\) from equation 102.91, which leads to the equation of skeleton equilibrium

\[
\sigma_{ij,j} - \rho \dot{u}_i + \rho b_i + n p_{i,i} + n \rho_f \ddot{u}_i - n \rho_f b_i = 0
\] (102.95)

By substituting \(\rho = (1 - n) \rho_s + n \rho_f\)

\[
\sigma_{ij,j} - (1 - n) \rho_s \ddot{u}_i - n \rho_f \ddot{u}_i + (1 - n) \rho_s b_i + n \rho_f b_i + n p_{i,i} + n R_i + n \rho_f \ddot{u}_i - n \rho_f b_i = 0
\]

\[
\sigma_{ij,j} + n p_{i,i} + (1 - n) \rho_s b_i - (1 - n) \rho_s \ddot{u}_i + n R_i = 0
\] (102.96)

By using the definition of effective stress, equation 102.83, equation 102.95 becomes

\[
\sigma''_{ij,j} - (\alpha - n) p_{i,i} + (1 - n) \rho_s b_i - (1 - n) \rho_s \ddot{u}_i + n R_i = 0
\] (102.97)

**Fluid Part Equilibrium Equation.** The fluid part equilibrium equation can be obtained simply by \([n \times (102.92)]\), i.e.

\[
-n p_{i,i} - n R_i - n \rho_f \ddot{u}_i - \rho_f \ddot{w}_i + n \rho_f b_i = 0
\]

\[
-n p_{i,i} - n R_i - n \rho_f (\ddot{u}_i + \frac{\ddot{w}_i}{n}) + n \rho_f b_i = 0
\] (102.98)

From equation (102.94),

\[
\ddot{U}_i = \ddot{u}_i + \frac{\ddot{w}_i}{n}
\] (102.99)

so that equation (102.98) becomes:

\[
-n p_{i,i} + n \rho_f b_i - n \rho_f \ddot{U}_i - n R_i = 0
\] (102.100)

**Mixture Balance of Mass.** By differentiating equation (102.94) in time and space

\[
\dot{w}_{i,i} = n \ddot{U}_{i,i} - n \dot{u}_{i,i}
\] (102.101)

Notice that \(\dot{\varepsilon}_{ii} = \ddot{u}_{i,i}\), so that equation (102.101) becomes

\[
\dot{w}_{i,i} = n \ddot{U}_{i,i} - n \dot{\varepsilon}_{ii}
\] (102.102)

By substituting (102.102) to (102.93)

\[
n \ddot{U}_{i,i} - n \dot{\varepsilon}_{ii} + \alpha \dot{\varepsilon}_{ii} + \frac{\dot{p}}{Q} = 0
\] (102.103)

or:
Developed is a set of modified governing equations (102.97), (102.100) and (102.104). They are summarized below

\[ -n\dot{U}_{i,i} = (\alpha - n)\dot{\varepsilon}_{ii} + \frac{1}{Q}\dot{p} \]  

(102.104)

\[ \sigma''_{ij,j} - (\alpha - n)p_{,i} + (1 - n)\rho_s b_i - (1 - n)\rho_s \ddot{u}_i + nR_i = 0 \]  

(102.105)

\[ -np_{,i} + n\rho_f b_i - n\rho_f \ddot{U}_i - nR_i = 0 \]  

(102.106)

\[ -n\dot{U}_{i,i} = (\alpha - n)\dot{\varepsilon}_{ii} + \frac{1}{Q}\dot{p} \]  

(102.107)

From the modified equation set (102.105), (102.106) and (102.107), it is noted that only \( \ddot{u}_i \) occurs in the first equation, and only \( \ddot{U}_i \) in the second, thus leading to a convenient diagonal form in discretization.

Obtained is a complete equation system given by (102.105), (102.106) and (102.107). With the basic definitions introduced earlier, there are three essential unknowns:

1. three solid displacement \( u_i \)
2. pore pressure \( p \)
3. three fluid displacement \( U_i \)

The boundary conditions imposed on these variables will complete the problem. These boundary conditions are:

- For the momentum balance part,
  - on boundary \( \Gamma_t \), traction \( t_i(t) (\text{or } \sigma_{ij} n_j) \), where \( n_i \) is the \( i \)-th component of the normal to the boundary.
  - On boundary \( \Gamma_u \), the displacement \( u_i \) is given.

- For the fluid part, again the boundary is divided into two parts:
  - On \( \Gamma_p \), the pressure \( p \) is specified,
  - on \( \Gamma_w \), the normal outflow \( \dot{w}_n \) is specified. For impermeable boundary a zero value for the outflow should be specified.
The boundary conditions can be summarized below

\[ \Gamma = \Gamma_t \cup \Gamma_u \]
\[ \tau_i = \sigma_{ij}n_j = \tau_i \quad \text{on} \quad \Gamma = \Gamma_t \]
\[ u_i = u_i \quad \text{on} \quad \Gamma = \Gamma_u \quad (102.108) \]

and

\[ \Gamma = \Gamma_p \cup \Gamma_w \]
\[ p = p \quad \text{on} \quad \Gamma = \Gamma_p \]
\[ n^T w = w_n \quad \text{on} \quad \Gamma = \Gamma_w \quad (102.109) \]

102.12.1.4 Numerical Solution of the u-p-U Governing Equations

The solutions to the problems governed by the modified governing equation set (102.105), (102.106) and (102.107) can be found by solving partial differential equations, which can be written as

\[ A\ddot{\Phi} + B\dot{\Phi} + L(\Phi) = 0 \quad (102.110) \]

where A, B are constant matrices, and L is an operator involving spatial differentials. The dot notation represents the time differentiation. Vector of dependent variables, \( \Phi \) represents the displacement \( u \) or the pore fluid pressure \( p \).

The finite element solution of a problem proceeds as follows.

1. Discretize or approximate the unknown functions \( \Phi \) by a finite set of parameters \( \Phi_k \) and shape function \( H_k \). They are specified in space dimensions. Thus

\[ \Phi \approx \Phi^h = \sum_{k=1}^{n} H_k \Phi_k \quad (102.111) \]

2. Insert the value of the approximating function \( \hat{\Phi} \) into the differential equations to obtain a residual, then a set of weighted residual equations can be written in the form

\[ \int_{\Omega} W_j^T (A\ddot{\Phi}^h + B\dot{\Phi}^h + L(\Phi^h)) d\Omega = 0 \quad (102.112) \]

In the finite element method, the weighting functions \( W_j \) are usually identical to the shape functions.
The solid displacement $u_i$, the pore pressure $p$, and the absolute fluid displacement $U_i$ can be approximated using shape functions and nodal values.

\[
\begin{align*}
  u_i &= H_u^i \pi_{Ki} \\
  p &= H_p^i \pi_K \\
  U_i &= H_U^i \pi_{Ki}
\end{align*}
\]  

(102.113)

where $H_u^i$, $H_p^i$, and $H_U^i$ are shape functions for solid displacement, pore pressure and fluid displacement, respectively, $\pi_{Ki}, \pi_K, \pi_{Ki}$ are nodal values of solid displacement, pore pressure and fluid displacement, respectively.

**Numerical Solution of solid part equilibrium equation.** To obtain the numerical solution of the first equation, premultiply equation 102.105 by $H_u^i$ and integrate over the domain.

First term of (102.105) becomes

\[
\int_{\Omega} H_u^i \sigma_{ij,j} d\Omega = \int_{\Gamma_t} H_u^i n_j \sigma_{ij} d\Gamma - \int_{\Omega} H_u^i \sigma_{ij,j} d\Omega = (f_1^u)_{Ki} - \int_{\Omega} H_u^i D_{ijkl} \varepsilon_{ml} d\Omega = (f_1^u)_{Ki} - K_{E^P}^{Ki} \pi_{Pm} = (f_1^u)_{Ki} - K_{E^P}^{Ki} \pi_{Lj} = (f_1^u)_{Ki} - K_{E^P} \pi
\]  

(102.114)

where $K_{E^P}$ is the stiffness matrix of the solid part, $n_i$ is the direction of the normal on the boundary.

Second term of (102.105) becomes

\[
- \int_{\Omega} H_u^i (\alpha - n_i)p_i d\Omega = - \int_{\Gamma_p} H_u^i (\alpha - n_i)p_i d\Gamma + \int_{\Omega} H_u^i (\alpha - n_i)p d\Omega = - \int_{\Gamma_p} H_u^i (\alpha - n_i)p_i d\Gamma + \int_{\Omega} H_u^i (\alpha - n_i)H_u^p d\Omega |\pi_M = - (f_4^u)_{Ki} + (G_1)_{Ki} \pi_M = - f_4^u + (G_1) \pi
\]  

(102.115)

Third term of (102.105) (solid body force) is then
\[ \int_\Omega H^u_K(1 - n)\rho_s b_i d\Omega = (f^u_i)_{Ki} \]  \hspace{1cm} (102.116)

Fourth term of (102.105) can be written as

\[- \int_\Omega H^u_K(1 - n)\rho_s \delta_{ij} \ddot{u}_j d\Omega = -[\int_\Omega H^u_K(1 - n)\rho_s \delta_{ij} H^u_L d\Omega] \ddot{u}_{Lj} = -(M_s)_{KijL} \ddot{u}_{Lj} = -M_s \dddot{u} \]  \hspace{1cm} (102.117)

where \( M_s \) is the mass matrix of solid part. By substituting equations (102.88) and (102.94), last term of (102.105) (Damping Matrix) becomes

\[ \int_\Omega H^u_K n R_i d\Omega = \int_\Omega H^u_K n k^{-1}_{ij} \dot{w}_j d\Omega \]
\[ = \int_\Omega H^u_K n^2 k^{-1}_{ij} \dot{U}_j d\Omega - \int_\Omega H^u_K n^2 k^{-1}_{ij} \dot{u}_j d\Omega \]
\[ = [\int_\Omega H^u_K n^2 k^{-1}_{ij} H^u_L d\Omega] \dddot{U}_{Lj} - [\int_\Omega H^u_K n^2 k^{-1}_{ij} H^u_L d\Omega] \dddot{u}_{Lj} \]
\[ = (C_2)_{KijL} \dddot{U}_{Lj} - (C_1)_{KijL} \dddot{u}_{Lj} \]
\[ = C_2 \dddot{U} - C_1 \dddot{u} \]  \hspace{1cm} (102.118)

Equation (102.105) becomes

\[ -K^{EP} \dddot{u} + f^u_1 - f^u_4 + G_1 \dddot{p} + f^u_5 + M_s \dddot{u} + C_2 \dddot{U} - C_1 \dddot{u} = \mathbf{f}_s \]  \hspace{1cm} (102.119)

or

\[ K^{EP} \dddot{u} - G_1 \dddot{p} - C_2 \dddot{U} + C_1 \dddot{u} + M_s \dddot{u} = \mathbf{f}_s \]  \hspace{1cm} (102.120)

where

\[ \mathbf{f}_s = f^u_1 - f^u_4 + f^u_5 \]  \hspace{1cm} (102.121)

and in index form

\[ K^{EP}_{KijL} - (G_1)_{KijL} \dddot{p}_L + (C_2)_{KijL} \dddot{U}_{Lj} - (C_1)_{KijL} \dddot{u}_{Lj} + (M_s)_{KijL} \dddot{u}_{Lj} = (\mathbf{f}_s)_{Ki} \]  \hspace{1cm} (102.122)
where

\[
K^{EP} = (K^{EP})_{KimP} = \int_{\Omega} H_{K,j}^P D_{ijml} H_{P,l}^n d\Omega
\]

\[
G_1 = (G_1)_{KiM} = \int_{\Omega} H_{K,i}^n (\alpha - n) H_{M,j}^p d\Omega
\]

\[
C_2 = (C_2)_{KijL} = \int_{\Omega} H_{K}^n n^2 k_{ij}^{-1} H_{L}^U d\Omega
\]

\[
C_1 = (C_1)_{KijL} = \int_{\Omega} H_{K}^n n^2 k_{ij}^{-1} H_{L}^U d\Omega
\]

\[
M_s = (M_s)_{KijL} = \int_{\Omega} H_{K}^n (1 - n) \rho_s \delta_{ij} H_{L}^U d\Omega
\]

\[
\bar{f} = (\bar{f}_s)_{Ki} = (f_s^U)_{Ki} - (f_s^U)_{Ki} + (f_s^\rho)_{Ki}
\]

(102.123)

**Numerical Solution of fluid part equilibrium equation.** From equations (102.88) and (102.94),

\[
R_i = \bar{n} k_{ij}^{-1} (\dot{U}_j - \dot{u}_j)
\]

(102.124)

By substituting (102.124) into equation (102.106),

\[
- np_{i,j} + n \rho_f b_i - n \rho_f \dot{U}_i - n^2 k_{ij}^{-1} (\dot{U}_j - \dot{u}_j) = 0
\]

(102.125)

By premultiplying (102.125) by \( H_{K}^U \) and integrating over the domain, first term of (102.125) becomes

\[
- \int_{\Omega} n H_{K,ij}^U p_{i,j} d\Omega = - \int_{\Gamma_p} n H_{K,ij}^U n_p d\Gamma + \int_{\Omega} n H_{K,ij}^U p d\Omega
\]

\[
= -(f_1)_{Ki} + [\int_{\Omega} n H_{K,ij}^U H_{M,j}^p d\Omega] \bar{p}_M
\]

\[
= -(f_1)_{Ki} + (G_2)_{KiM} \bar{p}_M
\]

\[
= -(f_1)_{Ki} + (G_2) \bar{p}
\]

(102.126)

Second term of (102.125) is then

\[
\int_{\Omega} H_{K,ij}^U \rho_f b_{i,j} d\Omega = (f_2)_{Ki}
\]

(102.127)

Third term of (102.125) (Lumped mass matrix obtained by multiplying \( \delta_{ij} \)) becomes

\[
- \int_{\Omega} H_{K,ij}^U n \rho_f \delta_{ij} \ddot{U}_j d\Omega = - [\int_{\Omega} H_{K,ij}^U n \rho_f \delta_{ij} H_{L,j}^U d\Omega] \ddot{U}_j
\]

\[
= -(M_f)_{KijL} \ddot{U}_j
\]

\[
= - M_f \ddot{U}
\]

(102.128)
Forth term of (102.125) becomes
\[- \int_{\Omega} H_K^{ij} k_{ij}^{-1} \hat{U}_j d\Omega + \int_{\Omega} H_K^{ij} k_{ij}^{-1} \hat{u}_j d\Omega = - \int_{\Omega} H_K^{ij} k_{ij}^{-1} H_L^{ij} d\Omega \hat{U}_j \]
\[+ \left[ \int_{\Omega} H_K^{ij} k_{ij}^{-1} H_L^{ij} d\Omega \right] \hat{u}_j \]
\[= - (C_3)_{KijL} \hat{U}_j + (C_2)_{LjiK} \hat{u}_j \]
\[= C_3 \hat{U} + C_2^T \hat{u} \] (102.129)

Equation (102.125) becomes
\[- f_1 + G_2 \hat{p} + f_2 - M_f \ddot{U} - C_3 \dot{U} + C_2^T \dot{u} = 0 \] (102.131)

or
\[- G_2 \hat{p} - C_2^T \dot{u} + C_3 \dot{U} + M_f \ddot{U} = \bar{f}_f \] (102.132)

where
\[\bar{f}_f = f_2 - f_1 \] (102.133)

and in index form
\[-(G_2)_{KiM} \hat{p}_M - (C_2)_{LjiK} \hat{u}_L \dot{u}_j + (C_3)_{KijL} \dot{U}_L \dot{U}_j + (M_f)_{KijL} \ddot{U}_L \dot{U}_j = (f_f)_{Ki} \] (102.134)

where
\[(f_f)_{Ki} = (f_1)_{Ki} - (f_2)_{Ki} \]
\[G_2 = (G_2)_{KiN} = \int_{\Omega} n H_K^{ij} H_M^{ij} d\Omega \]
\[C_2^T = (C_2^T)_{KijL} = \int_{\Omega} H_K^{ij} k_{ij}^{-1} H_L^{ij} d\Omega \]
\[C_3 = (C_3)_{KijL} = \int_{\Omega} H_K^{ij} k_{ij}^{-1} H_L^{ij} d\Omega \]
\[M_f = (M_f)_{KijL} = \int_{\Omega} H_K^{ij} \rho f \delta_{ij} H_L^{ij} d\Omega \] (102.135)

Numerical Solution of flow conservation equation. By integrating (102.107) in time and noticing that \( \varepsilon_{ii} = u_{i,i} \),
\[-nU_{i,i} = (\alpha - n) \varepsilon_{ii} + \frac{1}{Q^2} \rho \]
\[\bar{p}_M = (\alpha - n) \varepsilon_{ii} + \frac{1}{Q^2} \rho \] (102.136)

By multiplying (102.136) by \( H_M^{ij} \) and integrating over domain, first term of (102.136) becomes
\[- \int_{\Omega} H_M^{ij} H_L^{ij} d\Omega \hat{U}_L \dot{U}_j = -(G_2)_{MLj} \hat{U}_L \dot{U}_j = -G_2^T \hat{U} \] (102.137)
Second term of (102.136) is
\[
\int_{\Omega} H_{M}^p(\alpha - n)u_{i,i}d\Omega = \int_{\Omega} H_{M}^p(\alpha - n)H_{L,j}^M d\Omega \bar{u}_{Lj} = (G_1)_{LjM}\bar{u}_{Lj} = G_1^T\bar{u}
\] (102.138)

Third term of (102.136) becomes
\[
\int_{\Omega} H_{N}^p \frac{1}{Q} H_{M}^p d\Omega \bar{p}_N = P_{NM}M = P\bar{p}
\] (102.139)

The equation (102.136) becomes
\[
G_2^T\bar{U} + G_1^T\bar{u} + P\bar{p} = 0
\] (102.140)

in index form
\[
(G_2)_{LiK}\bar{U}_{Li} + (G_1)_{LiK}\bar{u}_{Li} + P_{KL}\bar{p}_L = 0
\] (102.141)

### 102.12.1.5 Matrix form of the governing equations.

The numerical forms of governing equations (102.120), (102.132) and (102.140) can be written together in the matrix form as
\[
\begin{bmatrix}
M_s & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & M_f
\end{bmatrix}
\begin{bmatrix}
\ddot{u} \\
\ddot{p} \\
\ddot{U}
\end{bmatrix}
+ \begin{bmatrix}
C_1 & 0 & -C_2 \\
0 & 0 & 0 \\
-C_2^T & 0 & C_3
\end{bmatrix}
\begin{bmatrix}
\dot{u} \\
\dot{p} \\
\dot{U}
\end{bmatrix}
+ \begin{bmatrix}
K^{EP} & -G_1 & 0 \\
-G_1^T & -P & -G_2^T \\
0 & -G_2 & 0
\end{bmatrix}
\begin{bmatrix}
\ddot{u} \\
\ddot{p} \\
\ddot{U}
\end{bmatrix}
= \begin{bmatrix}
\dddot{U}_s \\
0
\end{bmatrix}
\] (102.142)

or in index form
\[
\begin{bmatrix}
(M_s)_{KijL} & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & (M_f)_{KijL}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_{Lj} \\
\ddot{p}_N \\
\ddot{U}_{Lj}
\end{bmatrix}
+ \begin{bmatrix}
(C_1)_{KijL} & 0 & -(C_2)_{KijL} \\
0 & 0 & 0 \\
-(C_2)_{LjiK} & 0 & (C_3)_{KijL}
\end{bmatrix}
\begin{bmatrix}
\dot{u}_{Lj} \\
\dot{p}_N \\
\dot{U}_{Lj}
\end{bmatrix}
+ \begin{bmatrix}
(K^{EP})_{KijL} & -(G_1)_{KijM} & 0 \\
-(G_1)_{LjM} & -P_{MN} & -(G_2)_{LjM} \\
0 & -(G_2)_{KijL} & 0
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_{Lj} \\
\ddot{p}_M \\
\ddot{U}_{Lj}
\end{bmatrix}
= \begin{bmatrix}
\dddot{U}_{Ki}^{solid} \\
0
\end{bmatrix}
\] (102.143)
where
\[
M_s = (M_s)_{KijL} = \int_{\Omega} H^u_K (1 - n) \rho_s \delta_{ij} H^u_L d\Omega
\]
\[
M_f = (M_f)_{KijL} = \int_{\Omega} H^u_K n \rho_f \delta_{ij} H^u_L d\Omega
\]
\[
C_1 = (C_1)_{KijL} = \int_{\Omega} H^u_K n^2 k_{ij}^{-1} H^u_L d\Omega
\]
\[
C_2 = (C_2)_{KijL} = \int_{\Omega} H^u_K n^2 k_{ij}^{-1} H^U_L d\Omega
\]
\[
C_3 = (C_3)_{KijL} = \int_{\Omega} H^U_K n^2 k_{ij}^{-1} H^U_L d\Omega
\]
\[
K^{EP} = (K^{EP})_{KijL} = \int_{\Omega} H^u_K m D_{mn} H^u_{L,n} d\Omega
\]
\[
G_1 = (G_1)_{KiM} = \int_{\Omega} H^u_K \alpha (\alpha - n) H^p_M d\Omega
\]
\[
G_2 = (G_2)_{KiM} = \int_{\Omega} n H^u_K H^p_i d\Omega
\]
\[
P = P_{NM} = \int_{\Omega} H^p_N \frac{1}{Q} H^p_M d\Omega
\]

(102.144)

\[
\bar{f}^{\text{solid}}_{Ki} = (f^u_1)_{Ki} - (f^u_4)_{Ki} + (f^u_5)_{Ki}
\]
\[
\bar{f}^{\text{fluid}}_{Ki} = -(f^U_1)_{Ki} + (f^U_2)_{Ki}
\]
\[
(f^u_1)_{Ki} = \int_{\Gamma_t} H^u_K n_j \sigma''_{ij} d\Gamma
\]
\[
(f^u_2)_{Ki} = \int_{\Gamma_p} H^u_K (\alpha - n) n_i p d\Gamma
\]
\[
(f^u_3)_{Ki} = \int_{\Gamma_p} H^u_K (1 - n) \rho_s b_i d\Omega
\]
\[
(f^U_1)_{Ki} = \int_{\Gamma_p} n H^U_K n_i p d\Gamma
\]
\[
(f^U_2)_{Ki} = \int_{\Omega} n H^U_K \rho f b_i d\Omega
\]

(102.145)

Functions \(N^u, N^p, N^U\) are shape functions for unknown field of skeleton displacements, pore fluid pressures and fluid displacements, respectively, while \(\rho, \rho_s, \rho_f\) are the density of the total, solid and fluid phases, respectively, \(n\) is the porosity, and by its definition \(\rho = (1 - n) \rho_s + n \rho_f\), the symbol \(n_i\) is the direction of the normal on the boundary, \(u_i\) is the displacement of the solid part, \(p\) is pore fluid pressure and \(U_i\) is the absolute displacement of the fluid part. Equation (102.142) represents the general form \((u - p - U)\) for coupled system which can be written in a familiar form as
\[
M \ddot{x} + C \dot{x} + K x = f
\]

(102.146)
where \( x \) represents the generalized unknown variable. The solution of this equation for each time step will render an unknown field for given initial and boundary conditions.

### 102.12.1.6 Choice of shape functions

Isoparametric elements are used in previous sections, where the coordinates are interpolated using the same shape functions as for the unknown. This mapping allows using elements of more arbitrary shape than simple forms such as rectangles and triangles. But in static or dynamic undrained analysis the permeability (and compressibility) matrices are zero, i.e. \( Q \to \infty, \text{and } P \to 0 \), resulting in a zero-matrix diagonal term in the equation (102.143).

The matrix to be solved is the same as that in the solutions of problems of incompressible elasticity or fluid mechanics. Actually, a wide choice of shape functions is available if the limiting (undrained) condition is never imposed. Due to the presence of first derivatives in space in all the equations, it is necessary to use "\( C_\text{o} \)-continuous" interpolation functions and the suitable element forms are shown in Fig. 102.11.

![Figure 102.11: Shape functions used for coupled analysis, displacement \( u \) and pore pressure \( p \) formulation](image)
Figure 102.12: 8 node u-p-U brick element. Note that all seven DOFs (three porous solid displacements \( u_i \), pore fluid pressure \( p \) and pore fluid displacements \( U_i \) are defined at each node.

Figure 102.13: 20 node brick element. Note that all seven DOFs (three porous solid displacements \( u_i \), pore fluid pressure \( p \) and pore fluid displacements \( U_i \) are defined at each node.
Figure 102.14: 27 node brick element. Note that all seven DOFs (three porous solid displacements $u_i$, pore fluid pressure $p$ and pore fluid displacements $U_i$ are defined at each node.

102.12.1.7 8 Node $u-p-U$ Brick

102.12.1.8 20 Node $u-p-U$ Brick

102.12.1.9 27 Node $u-p-U$ Brick

102.12.2 $u-p-U$ Formulation for Partially Saturated, Unsaturated Material

Coming SOON, by end of Winter 2020

102.12.3 $u-p$ Formulation

102.12.3.1 Governing Equations of Porous Media

The formulation given here is based on Zienkiewicz et al. (1999b).

The first governing equation of porous media is total momentum balance equation:

$$\sigma_{ij,j} - \rho \ddot{u}_i + \rho b_i = 0$$  \hspace{1cm} (102.147)

where $\sigma_{ij} = \sigma''_{ij} - \alpha p \delta_{ij}$ and $\rho = (1 - n) \rho_s + n \rho_f$.

The second governing equation is the fluid mass balance equation:

$$\left(k_{ij} (-p_{,j} + \rho_f b_j)\right)_{,i} + \alpha \ddot{u}_{i,i} + \frac{\dot{p}}{Q_{sf}} = 0$$  \hspace{1cm} (102.148)
where
\[ k_{ij} = \frac{k'_{ij}}{g\rho_f} = \frac{k'_{ij}}{\gamma_f} \quad (102.149) \]
and \( k'_{ij} \) is the permeability in Darcy's law with the same unit as velocity.

\[ Q_{sf} = \frac{K_sK_f}{K_s + K_f} \quad (102.150) \]
is the total compression modulus, \( K_s \) and \( K_f \) are solid and fluid compression modulus, respectively.

The boundary conditions are
\[ \sigma_{ij} n_j = \bar{t}_i \quad \text{on} \quad \Gamma = \Gamma_t \quad (102.151) \]
\[ u_i = \bar{u}_i \quad \text{on} \quad \Gamma = \Gamma_u \quad (102.152) \]
\[ n_i w_i = n_i k_{ij} (-p_{ji} + \rho_f b_j) = \bar{w} = -\bar{q} \quad \text{on} \quad \Gamma = \Gamma_w \quad (102.153) \]
\[ p = \bar{p} \quad \text{on} \quad \Gamma = \Gamma_p \quad (102.154) \]
where \( \bar{w} \) is the outflow and \( \bar{q} \) is the influx.

### 102.12.3.2 Numerical Solutions of the Governing Equations

The solid displacement \( u_i \) and the pore pressure \( p \) can be approximated using shape functions and nodal values:
\[ u_i = N^u_K \bar{u}_K \quad (102.155) \]
\[ p = N^p_L \bar{p}_L \quad (102.156) \]

Similar approximations are applied to \( \dot{u}_i, \ddot{u}_i, \dot{p} \) and \( \ddot{p} \).

**Numerical solution of the total momentum balance** The numerical solution of the total momentum balance is
\[ \int_{\Omega} N^u_K (\sigma_{ij,j} - \rho \ddot{u}_i + \rho b_i) \, d\Omega = 0 \quad (102.157) \]
First-term of (102.157) becomes
\[
\int_{\Omega} N_K^u \sigma_{ij,j} d\Omega = \int_{\Gamma_t} N_K^u \sigma_{ij,j} n_j d\Omega - \int_{\Omega} N_{K,j}^u \sigma_{ij} d\Omega
\]
\[
= \int_{\Gamma_t} N_K^u \bar{\epsilon}_i d\Omega - \int_{\Omega} N_{K,j}^u (\sigma_{ij}'' - \alpha p \delta_{ij}) d\Omega
\]
\[
= (f_1^u)_{Ki} - \int_{\Omega} N_{K,j}^u \sigma_{ij,j} d\Omega + \int_{\Omega} N_{K_i}^u \alpha p d\Omega
\]
\[
= (f_1^u)_{Ki} - \int_{\Omega} N_{K,j}^u D_{ijml} \varepsilon_{ml} d\Omega + \int_{\Omega} \alpha N_{K,i}^u N_{N}^p d\Omega \bar{p}_N
\]
\[
= (f_1^u)_{Ki} - (K_{ep}^{ep}) \bar{u}_m + (Q_{KiN}) \bar{p}_N
\]
\[
= f_1^u - (K_{ep}^{ep}) \bar{\epsilon} + Q \bar{p}
\] (102.158)

Second term of (102.157) becomes
\[
- \int_{\Omega} N_K^u \rho \ddot{\epsilon}_i d\Omega = - \int_{\Omega} N_K^u \rho N_L^u d\Omega \ddot{\epsilon}_L
\]
\[
= - \int_{\Omega} N_K^u \rho N_L^u d\Omega \ddot{\epsilon}_L
\]
\[
= - \int_{\Omega} \alpha \dot{u}_i, i d\Omega
\]
\[
= -(M_{KijL}) \ddot{u}_L
\]
\[
= -M \ddot{u}
\] (102.159)

The third term of (102.157) becomes
\[
\int_{\Omega} N_K^u \rho b_i d\Omega = (f_2^u)_{Ki}
\]
\[
= f_2^u
\] (102.160)

The equation (102.157) thus becomes
\[
(M_{KijL}) \ddot{u}_L - (Q_{KiN}) \bar{p}_N + (K_{ep}^{ep}) \ddot{u}_L = (f_1^u)_{Ki} + (f_2^u)_{Ki} = (f^u)_{Ki}
\] (102.161)

or
\[
M \ddot{u} - Q \bar{p} + (K_{ep}^{ep}) \ddot{u} = f_1^u + f_2^u = f^u
\] (102.162)

Numerical solution of the fluid mass balance  The numerical solution of the fluid mass balance is
\[
\int_{\Omega} N_M^p \left( k_{ij} (-p_{ij} + \rho f_j b_j)_{,j} + \alpha \dot{u}_{i,i} + \frac{\dot{p}}{Q_{sf}} \right) d\Omega = 0
\] (102.163)
First term of (102.163) becomes
\[
\int \Omega N^p_M \left( k_{ij}(-p_j + \rho_f b_j) \right)_{ij} \, d\Omega
\]  
(102.164)
\[= \int_{\gamma_w} N^p_M \tilde{w} i_n \, d\Omega - \int \Omega N^p_{M,i} k_{ij}(-p_j + \rho_f b_j) \, d\Omega \]
\[= \left( f^p_1 \right)_M + \int \Omega N^p_{M,i} k_{ij}p_j \, d\Omega - \int \Omega N^p_{M,i} k_{ij}\rho_f b_j \, d\Omega \]
\[= \left( f^p_1 \right)_M + \int \Omega N^p_{M,i} k_{ij}p_j \, d\Omega - \left( f^p_2 \right)_M \]
\[= \left[ f^p_1 + H\tilde{p} - f^p_2 \right] \]  
(102.165)

Second term of (102.163) becomes
\[
\int \Omega N^p_M \alpha \dot{u}_{i,i} \, d\Omega = \left[ \int \Omega N^p_M \alpha N^p_{L,j} \, d\Omega \right] \tilde{u}_{Lj} \]
\[= \left( Q_{LM} \right) \tilde{u}_{Lj} \]
\[= Q^T \tilde{u} \]  
(102.166)

The third term of (102.163) becomes
\[
\int \Omega N^p_M \dot{p} \, d\Omega = \left[ \int \Omega N^p_M \frac{1}{Q_s} N^p_K \, d\Omega \right] \tilde{p}_N \]
\[= \left( S_{MN} \right) \tilde{p}_N \]  
(102.167)

The equation (102.163) thus becomes
\[\left( H_{MN} \right) \tilde{p}_N + \left( Q_{LM} \right) \tilde{u}_{Lj} + \left( S_{MN} \right) \tilde{p}_N = - \left( f^p_1 \right)_M + \left( f^p_2 \right)_M \]  
(102.168)

or
\[H\tilde{p} + Q^T \tilde{u} + S\tilde{p} = -f^p_1 + f^p_2 = f^p \]  
(102.169)

Matrix form of the governing equations
Combine equation (102.161) and (102.168), to obtain
\[
\begin{bmatrix}
M_{KiLj} & 0 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
\dot{u}_{Lj} \\
\tilde{p}_N
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 \\
Q_{LM} & S_{MN}
\end{bmatrix}
\begin{bmatrix}
\dot{u}_{Lj} \\
\tilde{p}_N
\end{bmatrix}
+ \begin{bmatrix}
K^{(p)}_{KiLj} & -Q_{KiN} \\
0 & H_{MN}
\end{bmatrix}
\begin{bmatrix}
u_{Lj} \\
p_N
\end{bmatrix}
= \begin{bmatrix}
f^p_{K_i} \\
f^p_M
\end{bmatrix}
\]  
(102.170)
or, by combining equations (102.162) and (102.169), obtain

$$\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{u} \\ \ddot{p} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ Q^T & S \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{p} \end{bmatrix} + \begin{bmatrix} K^{ep} & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f^u \\ f^p \end{bmatrix}$$ (102.171)

where

$$f^u \leftrightarrow f^u_K = (f^u_1)K_i + (f^u_2)K_i$$ (102.172)

$$f^p \leftrightarrow f^p_M = -(f^p_1)_M + (f^p_2)_M$$ (102.173)

and

$$f^u_1 \leftrightarrow (f^u_1)_K = \int_{\Gamma_t} N^u_K \bar{f}_i d\Gamma$$ (102.174)

$$f^u_2 \leftrightarrow (f^u_2)_K = \int_{\Omega} N^u_K \rho \bar{b}_i d\Omega$$ (102.175)

$$f^p_1 \leftrightarrow (f^p_1)_M = \int_{\Gamma_w} N^p_M \bar{w} d\Gamma$$ (102.176)

$$f^p_2 \leftrightarrow (f^p_2)_M = \int_{\Omega} \rho N^p_M k_{ij} \rho f_j f_i d\Omega$$ (102.177)

$$M \leftrightarrow M_{Kij} = \delta_{ij} \int_{\Omega} N^u_K \rho N^u_L d\Omega$$ (102.178)

$$Q \leftrightarrow Q_{KIN} = \int_{\Omega} \alpha N^u_K \rho N^p_N d\Omega$$ (102.179)

$$S \leftrightarrow S_{MN} = \int_{\Omega} N^p_M \frac{1}{Q_{sf}} N^p_N d\Omega$$ (102.180)

$$H \leftrightarrow H_{MN} = \int_{\Omega} N^p_{M,i} k_{ij} N^p_{N,j} d\Omega$$ (102.181)

102.12.3.3 8 Node $u-p$ Brick

102.13 Material and Geometric Non–Linear Finite Element Formu-lation

102.13.1 Introduction

Presented here is a detailed formulation of material and geometric non–linear static finite element system of equations. The configuration of choice is material or Lagrangian. Eulerian and mixed Eulerian–Lagrangian configuration will be mentioned as need be.
102.13.2 Equilibrium Equations

The local form of equilibrium equations in material format (Lagrangian) for static case can be written as:

\[ P_{i,J} - \rho_0 b_i = 0 \] (102.182)

where \( P_{i,J} = S_{I,J}(F_i)^t \) and \( S_{I,J} \) are first and second Piola–Kirchhoff stress tensors, respectively and \( b_I \) are body forces.

Weak form of equilibrium equations is obtained by premultiplying 102.182 with virtual displacements \( \delta u_i \) and integrating by parts on the initial configuration \( B_0 \) (initial volume \( V_0 \)):

\[ \int_{V_0} \delta u_{i,j} P_{i,j} dV = \int_{V_0} \rho_0 \delta u_i b_i dV - \int_{S_0} \delta u_{i,t} dS \] (102.183)

It proves beneficial to rewrite Lagrangian format of weak form of equilibrium equilibrium by using sym-
metric second Piola–Kirchhoff stress tensor $S_{ij}$:

$$
\int_{V_0} \delta u_{i,j} F_{jl} S_{il} dV = \\
\int_{V_0} \frac{1}{2} \left( \delta u_{i,j} F_{jl} + F_{lj} \delta u_{j,i} \right) S_{il} dV = \\
\int_{V_0} \frac{1}{2} \left( \delta u_{i,j} (\delta_{jl} + u_{j,l}) + (\delta_{lj} + u_{l,j}) \delta u_{j,i} \right) S_{il} dV = \\
\int_{V_0} \frac{1}{2} \left( (\delta u_{i,j} + \delta u_{l,i}) + (\delta u_{i,j} u_{j,l} + u_{l,j} \delta u_{j,i}) \right) S_{il} dV =
$$

(102.184)

where symmetry of $S_{il}$ was used, definition for deformation gradient $F_{ki} = \delta_{ki} + u_{k,i}$. In addition, conveniently defined was differential operator $\hat{E}_{il}(\delta u_i, u_i)$ as

$$
\hat{E}_{il}(\delta u_i, u_i) = \frac{1}{2} (\delta u_{l,i} + \delta u_{i,l}) + \frac{1}{2} (u_{l,j} \delta u_{j,i} + \delta u_{i,j} u_{j,l})
$$

(102.185)

### 102.13.3 Formulation of Non–Linear Finite Element Equations

Consider the motion of a general solid in a fixed, non-moving Cartesian coordinate system, as shown in Figure (102.16), and assume that the solid can experience large displacements, large strains, and nonlinear constitutive response. The aim is to evaluate the equilibrium positions of the complete solid at discrete time points $0, \Delta t, 2\Delta t, \ldots$, where $\Delta t$ is an increment in time. To develop the solution strategy, assume that the solutions for the static and kinematic variables for all time steps from 0 to time $t$ inclusive, have been obtained. The solution process for the next required equilibrium position corresponding to time $t + \Delta t$ is typical and would be applied repetitively until a complete solution path has been found. Hence, in the analysis one follows all particles of the solid in their motion, from the original to the final configuration of the solid. In so doing, a Lagrangian ( or material ) formulation of the problem was adopted.

Weak format of the equilibrium equations can be obtained by premultiplying 102.182 with virtual displacements $\delta u_i$ and integrating by parts. We obtain the virtual work equations in the Lagrangian format:

$$
\int_{V_0} \delta u_{i,j} P_{ij} dV = \int_{V_0} \rho_0 \delta u_i b_i dV - \int_{S_0} \delta u_i \tilde{t}_i dV
$$

(102.186)

Virtual work equations can also be written in terms of second Piola–Kirchhoff stress tensor $S_{IJ}$ as:

$$
\int_{V_0} \delta u_{i,j} F_{jl} S_{il} dV = \int_{V_0} \rho_0 \delta u_i b_i dV + \int_{S_0} \delta u_i \tilde{t}_i dV
$$

(102.187)
which after some algebraic manipulations, and after observing that $S_{IJ} = S_{JI}$ yields (SEE ABOVE!) By introducing a differential operator $\hat{E}(u_1, u_2)$ as:

$$\hat{E}_{i l}(u_i, u_i) = \frac{1}{2} (u_{i,l} + l_{i,l}) + \frac{1}{2} (u_{i,j} + u_{j,i})$$

Virtual work equation 102.185 can be written as:

$$\int_{V_0} \hat{E}_{i l}(\delta u_i, u_i) S_{i l} dV = \int_{S_0} \rho_0 \delta u_i b_i dV + \int_{S_i} \delta u_i t_i dV$$

or as:

$$W(\delta u_i, u_i^{(k)})^{int} + W^{ext}(\delta u_i) = 0$$

with:

$$W^{int}(\delta u_i, u_i^{(k)}) = \int_{\Omega_c} \hat{E}_{i j}(\delta u_i, u_i^{(k)}) S_{i j}^{(k)} dV$$

$$= \int_{\Omega_c} ((\delta u_{j,i} + \delta u_{i,j}) + (u_{j,r} \delta u_{r,i} + \delta u_{i,r} u_{r,j})) S_{i j}^{(k)} dV$$

$$W^{ext}(\delta u_i) = -\int_{\Omega_c} \rho_0 \delta u_i b_i dV - \int_{\partial \Omega_c} \delta u_i t_i dS$$

Figure 102.16: Motion of a solid in non-moving Cartesian coordinate system.
102.13.4 Computational Domain in Incremental Analysis

This chapter elaborates on the choice of Total Lagrangian (TL) formulations as a computational domain. In addition, a Newton-type procedure is chosen for satisfying equilibrium, i.e., virtual work for a given computational domain. Given the displacement field \( u_i^{(k)}(X_j) \), in iteration \( k \), the iterative change \( \delta u_i \)

\[
\begin{align*}
  u_i^{(k+1)} &= u_i^{(k)} + \Delta u_i \\
\end{align*}
\]

is obtained from the linearized virtual work expression

\[
W(\delta u_i, u_i^{(k+1)}) \simeq W(\delta u_i, u_i^{(k)}) + \Delta W(\delta u_i, \Delta u_i; u_i^{(k)})
\]

Here, \( W(\delta u_i, u_i^{(k)}) \) is the virtual work expression

\[
W(\delta u_i, u_i^{(k)}) = W(\delta u_i, u_i^{(k)})^{\text{int}} + W^{\text{ext}}(\delta u_i)
\]

with

\[
\begin{align*}
  W^{\text{int}}(\delta u_i, n+1_u^{(k)}) &= \int_{\Omega_c} \hat{E}_{ij}(\delta u_i, n+1_u^{(k)}) n+1_S^{(k)} dV \\
  W^{\text{ext}}(\delta u_i) &= -\int_{\Omega_c} \rho_0 \delta u_i n+1_b_i dV - \int_{\partial \Omega_c} \delta u_i n+1_t_i dS \\
\end{align*}
\]

and the \( \Delta W(\delta u_i, \Delta u_i; u_i^{(k)}) \) is the linearization of virtual work

\[
\begin{align*}
  \Delta W(\delta u_i, \Delta u_i; u_i^{(k)}) &= \lim_{\epsilon \to 0} \frac{\partial W(\delta u_i, u_i + \epsilon \Delta u_i)}{\partial \epsilon} \\
  &= \int_{\Omega_c} \hat{E}_{ij}(\delta u_i, u_i) dS_{ij} dV + \int_{\Omega_c} \Delta \hat{E}_{ij}(\delta u_i, u_i) S_{ij} dV \\
  &= \int_{\Omega_c} \hat{E}_{ij}(\delta u_i, u_i) L_{ijkl} \hat{E}_{kl}(\Delta u_i, u_i) dV + \int_{\Omega_c} \Delta \hat{E}_{ij}(\delta u_i, u_i) S_{ij} dV \\
\end{align*}
\]

Here it was used that \( dS_{ij} = 1/2 L_{ijkl} dC_{kl} = L_{ijkl} \hat{E}_{kl}(\Delta u_i, u_i) \).

In order to obtain expressions for stiffness matrix, work is done on equation \( 102.198 \) in some more details. To this end, equation \( 102.198 \) can be rewritten by expanding definitions for \( \hat{E} \) as

\[
\begin{align*}
  \Delta W(\delta u_i, \Delta u_i; u_i^{(k)}) &= \\
  &= \frac{1}{4} \int_{\Omega_c} \left( (\delta u_{j,i} + \delta u_{i,j}) + (u_{j,s} \delta u_{r,i} + \delta u_{i,s} \delta u_{r,j})) L_{ijkl} \left( (\Delta u_{k,l} + \Delta u_{l,k}) + (u_{k,s} \Delta u_{s,l} + \Delta u_{l,s} u_{s,k}) \right) dV + \\
  &\quad + \int_{\Omega_c} \frac{1}{2} (\Delta u_{j,i} \delta u_{i,j} + \delta u_{i,l} \Delta u_{i,j}) S_{ij} dV \\
\end{align*}
\]
Or, by conveniently splitting the above equation

$$\Delta^1 W(\delta u_i, \Delta u_i; u_i^{(k)}) =$$

$$\frac{1}{4} \int_{\Omega_c} \left( (\delta u_{j,i} + \delta u_{i,j}) + (u_{j,r} \delta u_{r,i} + \delta u_{i,r} u_{r,j}) \right) L_{ijkl} \left( (\Delta u_{k,l} + \Delta u_{l,k}) + (u_{k,s} \Delta u_{s,l} + \Delta u_{l,s} u_{s,k}) \right) dV$$

(102.200)

$$\Delta^2 W(\delta u_i, \Delta u_i; u_i^{(k)}) = \int_{\Omega_c} \frac{1}{2} (\delta u_{j,i} + \delta u_{i,j}) S_{ij} dV$$

(102.201)

By further working on equation 102.200

$$\Delta^1 W(\delta u_i, \Delta u_i; u_i^{(k)}) = \int_{\Omega_c} \left( \frac{1}{2} (\delta u_{j,i} + \delta u_{i,j}) \right) L_{ijkl} \left( \frac{1}{2} (\Delta u_{k,l} + \Delta u_{l,k}) \right) dV$$

$$+ \int_{\Omega_c} \left( \frac{1}{2} (\delta u_{j,i} + \delta u_{i,j}) \right) L_{ijkl} \frac{1}{2} \left( \Delta u_{k,s} \Delta u_{s,l} + \Delta u_{l,s} u_{s,k} \right) dV$$

$$+ \int_{\Omega_c} \frac{1}{2} (u_{j,r} \delta u_{r,i} + \delta u_{i,r} u_{r,j}) L_{ijkl} \frac{1}{2} (u_{k,s} \Delta u_{s,l} + \Delta u_{l,s} u_{s,k}) dV$$

$$+ \int_{\Omega_c} \frac{1}{2} (u_{j,r} \delta u_{r,i} + \delta u_{i,r} u_{r,j}) L_{ijkl} \frac{1}{2} (\Delta u_{k,l} + \Delta u_{l,k}) dV$$

(102.202)

It should be noted that the Algorithmic Tangent Stiffness (ATS) tensor $L_{ijkl}$ poses both minor symmetries ($L_{ijkl} = L_{jikl} = L_{ijlk}$). However, Major symmetry cannot be guaranteed. Non–associated flow rules in elastoplasticity lead to the loss of major symmetry ($L_{ijkl} \neq L_{klij}$). Moreover, it can be shown (i.e., Jeremić and Sture (1997)) that there is algorithmic induced symmetry loss even for associated flow rules.

With the minor symmetry of $L_{ijkl}$ one can write (102.202) as:

$$\Delta^1 W(\delta u_i, \Delta u_i; u_i^{(k)}) = \int_{\Omega_c} \delta u_{i,j} L_{ijkl} \Delta u_{i,k} dV$$

$$+ \int_{\Omega_c} \delta u_{i,j} L_{ijkl} u_{k,s} \Delta u_{l,s} dV$$

$$+ \int_{\Omega_c} \delta u_{i,r} u_{r,j} L_{ijkl} u_{k,s} \Delta u_{l,s} dV$$

$$+ \int_{\Omega_c} \delta u_{i,r} u_{r,j} L_{ijkl} \Delta u_{i,k} dV$$

(102.203)

Similarly, by observing symmetry of second Piola–Kirchhoff stress tensor $S_{ij}$

$$\Delta^2 W(\delta u_i, \Delta u_i; u_i^{(k)}) = \int_{\Omega_c} \delta u_{i,i} \Delta u_{i,j} S_{ij} dV$$

(102.204)

Weak form of equilibrium expressions (i.e. (102.192) and (102.192) ) for internal ($W^{int}$) and external ($W^{ext}$) virtual work, with the above mentioned symmetry of $S_{ij}$ can be written as

$$W^{int}(\delta u_i, n_0 u_i^{(k)}) = \int_{\Omega_c} \delta u_{i,j} S_{ij} dV + \int_{\Omega_c} \delta u_{i,r} u_{r,j} S_{ij} dV$$

(102.205)
\[ W^{\text{ext}}(\delta u_i) = - \int_{\Omega_e} \rho_0 \delta u_i b_i \, dV - \int_{\partial \Omega_e} \delta u_i \, t_i \, dS \quad (102.206) \]

Standard finite element discretization of displacement field yields:

\[ u_i \approx \hat{u}_i = H_I \bar{u}_I \quad (102.207) \]

where \( \hat{u}_i \) is the approximation to exact, analytic (if it exists) displacement field \( u_i \), \( H_I \) are standard FEM shape functions and \( \bar{u}_I \) are nodal displacements. With this approximation

\[
\Delta^3 W(\delta u_i, \Delta u_i; u_i^{(k)}) = \int_{\Omega_e} (H_{I,j} \delta \bar{u}_I) \, L_{ijkl} \, (H_{Q,k} \Delta \bar{u}_Q) \, dV \\
+ \int_{\Omega_e} (H_{I,j} \delta \bar{u}_I) \, L_{ijkl} \, (H_{J,k} \bar{u}_J) \, (H_{Q,s} \Delta \bar{u}_Q) \, dV \\
+ \int_{\Omega_e} (H_{I,r} \delta \bar{u}_I) \, (H_{J,j} \bar{u}_J) \, L_{ijkl} \, (H_{J,k} \bar{u}_J) \, (H_{Q,s} \Delta \bar{u}_Q) \, dV \\
+ \int_{\Omega_e} (H_{I,r} \delta \bar{u}_I) \, (H_{J,j} \bar{u}_J) \, L_{ijkl} \, (H_{Q,k} \Delta \bar{u}_Q) \, dV \\
(102.208) \]

\[
\Delta^3 W(\delta u_i, \Delta u_i; u_i^{(k)}) = \int_{\Omega_e} (H_{I,l} \delta \bar{u}_I) \, (H_{Q,s} \Delta \bar{u}_Q) \, S_{ij} \, dV \\
(102.209) \]

\[
W^{\text{int}}(\delta u_i; n_i^{(k)}) = \int_{\Omega_e} (H_{I,j} \delta \bar{u}_I) \, S_{ij} \, dV + \int_{\Omega_e} (H_{I,r} \delta \bar{u}_I) \, (H_{J,j} \bar{u}_J) \, S_{ij} \, dV \\
(102.210) \]

\[
W^{\text{ext}}(\delta u_i) = - \int_{\Omega_e} \rho_0 \, (H_I \delta \bar{u}_I) \, b_i \, dV - \int_{\partial \Omega_e} (H_I \delta \bar{u}_I) \, t_i \, dS \\
(102.211) \]

Upon noting that virtual nodal displacements \( \delta u_I \) are any non–zero, continuous displacements, and since they occur in all expressions for linearized virtual work (from Equations \(102.194\), \(102.195\), \(102.196\), \(102.197\) and \(102.198\)) they can be factored out so that (while remembering that \( \Delta W^1 + \Delta W^2 + W^{\text{ext}} + W^{\text{int}} = 0 \)):

\[
\int_{\Omega_e} (H_{I,j}) \, L_{ijkl} \, (H_{Q,k} \Delta \bar{u}_Q) \, dV \\
+ \int_{\Omega_e} (H_{I,j}) \, L_{ijkl} \, (H_{J,k} \bar{u}_J) \, (H_{Q,s} \Delta \bar{u}_Q) \, dV \\
+ \int_{\Omega_e} (H_{I,r}) \, (H_{J,j} \bar{u}_J) \, L_{ijkl} \, (H_{J,k} \bar{u}_J) \, (H_{Q,s} \Delta \bar{u}_Q) \, dV \\
+ \int_{\Omega_e} (H_{I,r}) \, (H_{J,j} \bar{u}_J) \, L_{ijkl} \, (H_{Q,k} \Delta \bar{u}_Q) \, dV \\
+ \int_{\Omega_e} (H_{I,l}) \, (H_{Q,s} \Delta \bar{u}_Q) \, S_{ij} \, dV \\
+ \int_{\Omega_e} (H_{I,j}) \, S_{ij} \, dV + \int_{\Omega_e} (H_{I,r}) \, (H_{J,j} \bar{u}_J) \, S_{ij} \, dV \\
= \int_{\Omega_e} \rho_0 \, (H_I) \, b_i \, dV + \int_{\partial \Omega_e} (H_I) \, t_i \, dS \\
(102.212) \]
By rearranging previous equations, one can write:
\[
\begin{align*}
\left( \int_{\Omega_c} H_{I,J} L_{ijk} H_{Q,k} dV + \int_{\Omega_c} H_{I,J} L_{ijkl} H_{Q,k} dV + \int_{\Omega_c} H_{I,J} \tilde{u}_{Jr} L_{ijkl} H_{Q,k} dV \\
+ \int_{\Omega_c} H_{I,J} \tilde{u}_{Jr} L_{ijkl} H_{Q,k} dV + \int_{\Omega_c} H_{I,J} \tilde{u}_{Jr} L_{ijkl} H_{Q,s} dV \right) \Delta \bar{u}_Q \\
+ \int_{\Omega_c} (H_{I,J}) S_{ij} dV + \int_{\Omega_c} (H_{I,J}) (H_{I,J}) S_{ij} dV \\
= \int_{\Omega_c} \rho_0 (H_I) b_i dV + \int_{\partial\Omega_c} (H_I) t_i dS
\end{align*}
\]
(102.213)

The vectors of external and internal forces are
\[
\begin{align*}
f_{int} &= \frac{\partial(W^{int}(\delta u_i, n+1 u_i^{(k)}))}{\partial(\delta u_i)} \\
f_{ext} &= \frac{\partial(W^{ext}(\delta u_i))}{\partial(\delta u_i)}
\end{align*}
\]
(102.214)
(102.215)

The Algorithmic Tangent Stiffness (ATS) tensor \(L_{ijkl}^{ATS}\) is defined as a linearization of second Piola–Kirchhoff stress tensor \(S_{ij}\) with respect to the right deformation tensor \(C_{kl}\)
\[
dS_{ij} = \frac{1}{2} L_{ijkl} dC_{kl}
\]
with \(dC_{kl} = 2 \bar{E}_{kl}(du_i, u_i)\)
(102.216)

Then, the global algorithmic tangent stiffness matrix (tensor) is given as
\[
K_t = \frac{\partial(\Delta W(\delta u_i, \Delta u_i; u_i^{(k)}))}{\partial(\delta u_i)}
\]
(102.217)

The iterative change in displacement vector \(\Delta u_i\) is obtained by setting a linearized virtual work to zero
\[
W(\delta u_i, u_i^{(k+1)}) = 0 \Rightarrow W(\delta u_i, u_i^{(k)}) = -\Delta W(\delta u_i, \Delta u_i; u_i^{(k)})
\]
(102.218)

### 102.13.4.1 Total Lagrangian Format

The undeformed configuration \(\Omega_0\) is chosen as the computational domain \((\Omega_c = \Omega_0)\). The iterative displacement \(\Delta u_i\) is obtained from the equation
\[
W(\delta u_i, n+1 u_i^{(k)}) = -\Delta W(\delta u_i, \Delta u_i; n+1 u_i^{(k)})
\]
(102.219)

where
\[
W(\delta u_i, n+1 u_i^{(k)}) = \int_{\Omega_c} \bar{E}_{ij}(\delta u_i, n+1 u_i^{(k)}) n+1 S_{ij}^{(k)} dV \\
- \int_{\Omega_c} \rho_0 \delta u_i n+1 b_i dV - \int_{\partial\Omega_c} \delta u_i n+1 t_i dS
\]
(102.220)
and
\[
\Delta W(\delta u_i, \Delta u_i; n+1u_i^{(k)}) = \int_{\Omega_c} \hat{E}_{ij}(\delta u_i, n+1u_i^{(k)}) n+1L_{ijkl} \hat{E}_{kl}(\Delta u_i, n+1u_i^{(k)}) \, dV \\
+ \int_{\Omega_c} d\hat{E}_{ij}(\delta u_i, \Delta u_i) n+1S_{ij} \, dV
\] (102.221)

In the case of hyperelastic–plastic response, second Piola–Kirchhoff stress \( n+1S_{ij}^{(k)} \) is obtained by integrating the constitutive law, described in Chapter 106.4. It should be noted that by integrating in the intermediate configuration, obtained is Mandel stress \( n+1\bar{T}_{ij} \) and subsequently\(^{24} \) the second Piola–Kirchhoff stress \( \bar{S}_{kj} \). The ATS tensor \( \bar{L}_{ijkl} \) is then obtained based on \( \bar{S}_{kj} \). In order to obtain second Piola–Kirchhoff stress \( S_{kj} \) and ATS tensor in initial configuration, a pull-back from the intermediate configuration to the initial configuration is performed

\[
n+1S_{ij} = n+1F_{ip}^{p} n+1F_{jq}^{p} n+1\bar{S}_{pq}
\] (102.222)

\[
n+1L_{ijkl} = n+1F_{im}^{p} n+1F_{jn}^{p} n+1F_{kr}^{p} n+1F_{ls}^{p} n+1\bar{L}_{mnrs}
\] (102.223)

102.13.5 Finite Element Formulations

Presented here is a slightly different approach to developing large deformation FEM in total Lagrangian form. Lower case indices are used for variables in current configuration, while the capital case indices are used for the reference configuration.

102.13.5.1 Strong Form

The static, strong form of momentum balance in the current configuration is

\[
\frac{\partial\sigma_{ij}}{\partial x_j} + \rho b_i = 0
\] (102.224)

where \( \sigma_{ij} \) is the Cauchy stress, \( \rho \) is the material density, \( b_i \) is the material body force.

Used here was the so-called Total Lagrangian formulation that is based on the reference configuration.

The strong form of momentum balance can be expressed in the reference configuration

\[
\frac{\partial P_{iJ}}{\partial X_j} + \rho_0 b_i = 0
\] (102.225)

where \( P_{iJ} \) is the first Piola-Kirchhoff stress, \( \rho_0 \) is the material Lagrangian density and \( \rho_0 = J \rho \).

\(^{24}\bar{S}_{kj} = (\bar{C}_{uk})^{-1} \bar{T}_{ij} \)
102.13.5.2 Weak Form

The corresponding weak form of Equation 102.225 can be expressed as

\[ \int_{\Omega_0} \delta u_i \left( \frac{\partial P_{iJ}}{\partial X_J} + \rho_0 b_i \right) dV = 0 \]  

(102.226)

where \( \delta u_i \) is some arbitrary virtual displacement, \( \Omega_0 \) is the concerned domain of the reference configuration. Using the partial integration rule, the above equation can be alternatively expressed as

\[ \int_{\Omega_0} \frac{\partial}{\partial X_J} (P_{iJ} \delta u_i) dV - \int_{\Omega_0} P_{iJ} \frac{\partial \delta u_i}{\partial X_J} dV + \int_{\Omega_0} \rho_0 b_i \delta u_i dV = 0 \]  

(102.227)

The first term of Equation 102.227 can be rewritten in terms of the surface traction

\[ \int_{\Omega_0} \frac{\partial}{\partial X_J} (P_{iJ} \delta u_i) dV = \int_{\partial \Omega_0} t_i \delta u_i dA \]  

(102.228)

with the traction \( t_i = P_{iJ} H_J \), where \( H_J \) the unit surface normal vector in the reference configuration, and \( \partial \Omega_0 \) is the boundary of the reference domain \( \Omega_0 \).

The second term of Equation 102.227 can be rewritten as

\[ \int_{\Omega_0} P_{iJ} \frac{\partial \delta u_i}{\partial X_J} dV = \int_{\Omega_0} P_{iJ} \delta F_{iJ} dV = \int_{\Omega_0} S_{IJ} \delta E_{IJ} dV \]  

(102.229)

where \( S_{IJ} \) is the second Piola-Kirchhoff stress and \( E_{IJ} \) is the Lagrangian-Green strain. \( \delta F_{iJ} = \partial \delta u_i / \partial X_J \) is used.

The overall weak form in the reference configuration is now

\[ \int_{\Omega_0} S_{IJ} \delta E_{IJ} dV = \int_{\partial \Omega_0} t_i \delta u_i dA + \int_{\Omega_0} \rho_0 b_i \delta u_i dV \]  

(102.230)

102.13.5.3 Linearized Form

To utilize the iterative algorithm for incremental strategy, needed is the linearized form the governing equation 102.230.

The first term linearization of Equation 102.230 is

\[ \Delta \int_{\Omega_0} S_{IJ} \delta E_{IJ} = \int_{\Omega_0} \left[ \Delta S_{IJ} \delta E_{IJ} + S_{IJ} \delta (\Delta E_{IJ}) \right] dV \]  

\[ = \int_{\Omega_0} \left[ L_{IJKL} \Delta E_{KL} \delta E_{IJ} + S_{IJ} \delta (\Delta E_{IJ}) \right] dV \]  

(102.231)

where \( L_{IJKL} \) is the Lagrangian stiffness linked the second Piola-Kirchhoff stress \( S_{IJ} \) and the Lagrangian-Green strain \( E_{KL} \) by the relation

\[ S_{IJ} = L_{IJKL} E_{KL} \]  

(102.232)
The linearization of $E_{KL}$ is
\[
\Delta E_{KL} = \text{Sym} \left( F_K \frac{\partial \Delta u_b}{\partial X_L} \right)
\]  
(102.233)
where $\text{Sym}$ is the operator of tensor symmetry, defined as $\text{Sym}(A_{ij}) := (1/2)(A_{ij} + A_{ji})$. Similarly,
\[
\delta E_{IJ} = \text{Sym} \left( F_{aI} \frac{\partial \delta u_a}{\partial X_J} \right)
\]  
(102.234)
and
\[
\delta \left( \Delta E_{IJ} \right) = \text{Sym} \left( \frac{\partial \delta u_c}{\partial X_I} \frac{\partial \Delta u_c}{\partial X_J} \right)
\]  
(102.235)
Note that $S_{IJ}$ is a symmetric tensor, and $L_{IJKL}$ is a tensor with major and minor symmetries, Equation 102.231 can be expressed as
\[
\Delta \int_{\Omega_0} S_{IJ} \delta E_{IJ} = \int_{\Omega_0} \left[ \mathcal{L}_{IJKL} \left( F_{aI} \frac{\partial \delta u_a}{\partial X_J} \right) \left( F_{aI} \frac{\partial \delta u_a}{\partial X_J} \right) + S_{IJ} \left( \frac{\partial \delta u_c}{\partial X_I} \frac{\partial \Delta u_c}{\partial X_J} \right) \right] dV
\]  
(102.236)

The overall linearization form is thus
\[
\int_{\Omega_0} \left[ \frac{\partial \delta u_a}{\partial X_J} \left( F_{aI} F_{bK} \mathcal{L}_{IJKL} + \delta_{ab} S_{IJL} \right) \frac{\partial \Delta u_b}{\partial X_L} \right] dV = \int_{\partial \Omega_0} \Delta t_i \delta u_i dA + \int_{\Omega_0} \rho_0 \Delta b_i \delta u_i dV
\]  
(102.237)

### 102.13.5.4 Finite Element Form

In finite element form, the displacements $u_i$ are interpolated from the element nodal displacements $\bar{u}_{Ai}$:
\[
u_i = H_A \bar{u}_{Ai}
\]  
(102.238)
where $H_A$ is the element shape function of the node $A$, $\bar{u}_{Ai}$ is the node $A$ displacements, and
\[
\frac{\partial \delta u_a}{\partial X_J} = \frac{\partial H_A}{\partial X_J} \delta \bar{u}_{Aa}, \quad \frac{\partial \Delta u_b}{\partial X_L} = \frac{\partial H_B}{\partial X_L} \Delta \bar{u}_{Bb}
\]  
(102.239)
Equation 102.237 can be expressed as
\[
(K_{AaBb} \Delta \bar{u}_{Bb} - \Delta f^{ex}_{Aa}) \delta \bar{u}_{Aa} = 0
\]  
(102.240)
where
\[
K_{AaBb} = \int_{\Omega_0} \frac{\partial H_A}{\partial X_J} \left( F_{aI} F_{bK} \mathcal{L}_{IJKL} + \delta_{ab} S_{IJL} \right) \frac{\partial H_B}{\partial X_L} dV
\]  
(102.241)
\[
\Delta f^{ex}_{Aa} = \int_{\partial \Omega_0} H_A \Delta t_i dA + \int_{\Omega_0} \rho_0 H_A \Delta b_i dV
\]  
(102.242)
Due to the arbitrariness of the virtual nodal displacements, the expression in the parentheses should be zero in Equation 102.240, which gives the incremental finite element form:
\[
K_{AaBb} \Delta \bar{u}_{Bb} = \Delta f^{ex}_{Aa}
\]  
(102.243)
Chapter 103

Micromechanical Origins of Elasto-Plasticity

(1994-2002-2010-2019-2021-)
103.1 Chapter Summary and Highlights

This chapter is based in large part on lecture notes by Prof. Stein Sture (Sture, 1993).

103.2 Friction

103.2.1 Early Works

- Leonardo da Vinci, (1452-1519), worked and wrote about friction in 1493... (https://en.wikipedia.org/wiki/Leonardo_da_Vinci)
- Guillaume Amontons (1663-1705), Law of friction, rediscovered in 1699 (since da Vinci’s notes were lost), (https://en.wikipedia.org/wiki/Guillaume_Amontons)

Surface with asperities
grain assemblies, loose and dense
Saw-Teeth model analog
Frictional response of solids, friction angle for the polished mineral and dilatancy angle
Particle shapes
Particle rotations
types of packaging of particles

103.3 Particle Contact Mechanics

103.3.1 Particle Contact Mechanics, Axial Behavior

- Hertz contact theory, elastic (1885)
- Cattaneo and Midnlin theories, 1938 elastic and plastic contact

Two equal particles in normal contact
equations for $\alpha$, $\Delta$, $\sigma_N$, $\sigma_{N}^{max}$
Two unequal particles in normal contact
average normal stress and average normal strain
tangent stiffness, function of $\sqrt[3]{\sigma_{avg}}$
Bulk modulus for loose and dense packaging
Mindlin and Deresiewicz (1953)
Hashin (1983)
Rowe (1962)
Cosserat (1909)

103.3.2 Particle Contact Mechanics, Shear Behavior

Shear Behavior
  closed form solution for no-slip behavior
  equilibrium, integral of $\tau$
  slip ring
  SLIP ring
There is NO elastic behavior of particles in contact if there is ANY small amount of shear!
Example values for typical contact stress parameters
typical stiffness parameters for sands

103.4 Dilatancy
Chapter 104

Small Deformation Elasto-Plasticity

(In collaboration with Prof. Zhaohui Yang, Dr. Zhao Cheng, Dr. Nima Tafazzoli, Dr. Federico Pisanò, and Dr. Han Yang)
104.1 Chapter Summary and Highlights

104.2 Elasto–plasticity

104.2.1 Constitutive Relations for Infinitesimal Plasticity

A wide range of elasto–plastic materials can be characterized by means of a set of incremental constitutive relations of the general form:

\[
d\varepsilon_{ij} = d\varepsilon^e_{ij} + d\varepsilon^p_{ij} \tag{104.1}
\]

\[
d\sigma_{ij} = E_{ijkl} d\varepsilon^e_{kl} \tag{104.2}
\]

\[
d\varepsilon^p_{ij} = d\lambda \frac{\partial Q}{\partial \sigma_{ij}} = d\lambda m_{ij}(\sigma_{ij}, q_*) \tag{104.3}
\]

\[
dq_* = d\lambda h_*(\tau_{ij}, q_*) \tag{104.4}
\]

where, following standard notation \(\varepsilon_{ij}, \varepsilon^e_{ij}\) and \(\varepsilon^p_{ij}\) denotes the total, elastic and plastic strain tensor, (and \(d\varepsilon_{ij}\) is an increment of a strain tensor \(\varepsilon_{ij}\)), \(\sigma_{ij}\) is the Cauchy stress tensor, and \(q_*\) signifies some suitable set of internal variables\(^1\). The asterisk in the place of indices in \(q_*\) replaces \(n\) indices\(^2\). Equation (104.1) expresses the commonly assumed additive decomposition of the infinitesimal strain tensor into elastic and plastic parts. Equation (104.2) represents the generalized Hooke’s law\(^3\) which linearly relates stresses and elastic strains through a stiffness modulus tensor \(E_{ijkl}\). Equation (104.3) expresses a generally associated or non-associated flow rule for the plastic strain and (104.4) describes a suitable set of hardening laws, which govern the evolution of the plastic variables. In these equations, \(m_{ij}\) is the plastic flow direction, \(h_*\) the plastic moduli and \(d\lambda\) is a plastic parameter to be determined with the aid of the loading—unloading criterion, which can be expressed in terms of the Karush–Kuhn–Tucker condition (Karush, 1939; Kuhn and Tucker, 1951) as:

\[
F(\sigma_{ij}, q_*) \leq 0 \tag{104.5}
\]

\(^1\)In the simplest models of plasticity the internal variables are taken as either plastic strain components \(\varepsilon^p_{ij}\) or the hardening variables \(\kappa\) defined, for example as a function of inelastic (plastic) work, i.e. \(\kappa = f(W^p)\). See Lubliner (1990) page 115.

\(^2\) for example \(i, j\) if the variable is \(\varepsilon^p_{ij}\), or nothing if the variable is a scalar value, i.e. \(\kappa\)

\(^3\) also Eq. 104.157
\[ d\lambda \geq 0 \quad (104.6) \]

\[ F \, d\lambda = 0 \quad (104.7) \]

In the previous equations \( F(\sigma_{ij}, q^*) \) denotes the yield function of the material and (104.5) characterizes the corresponding elastic domain, which is presumably convex. Along any process of loading, conditions (104.5), (104.6) and (104.7) must hold simultaneously. For \( F < 0 \), equation (104.7) yields \( d\lambda = 0 \), i.e. elastic behavior, while plastic flow is characterized by \( d\lambda > 0 \), which with (104.7) is possible only if the yield criterion is satisfied, i.e. \( F = 0 \). From the latter constraint, in the process of plastic loading the plastic consistency conditions\(^4\) is obtained in the form:

\[ dF = \frac{\partial F}{\partial \sigma_{ij}} d\sigma_{ij} + \frac{\partial F}{\partial q^*} dq^* = n_{ij} d\sigma_{ij} + \xi^* dq^* = 0 \quad (104.8) \]

where:

\[ n_{ij} = \frac{\partial F}{\partial \sigma_{ij}} \quad (104.9) \]

\[ \xi^* = \frac{\partial F}{\partial q^*} \quad (104.10) \]

Equation (104.8) has the effect of confining the stress trajectory to the yield surface\(^5\). It is worthwhile noting that \( n_{ij} \) and \( \xi^* \) are normals to the yield surface in stress space and the plastic variable space respectively.

An interesting alternative way of representing non–associated flow rules can be found in Runesson (1987). A fictitious plastic strain derived from associated flow rule, \( e_{ij}^P \) is introduced. This fictitious plastic strain is assumed to be related to the real plastic strain \( \epsilon_{ij}^P \), which is derived from a non–associated flow rule\(^6\) through the linear transformation:

\[ e_{ij}^P = A_{ijkl} \epsilon_{kl}^P \quad (104.11) \]

Linear transformation tensor \( A_{ijkl} \) may be state dependent in general case, and it reduces to the symmetric part of the fourth order identity tensor\(^7\) for the case of associated plasticity.

---

\(^4\)first order accuracy condition.

\(^5\)Since it is only linear expansion stress trajectory is confined to the tangential plane only.

\(^6\)as in equation 104.3.

\(^7\)\( A_{ijkl} \equiv I_{ijkl}^{sym} = \frac{1}{2} (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) \).
It is often of interest to model deviatoric strains by an associated flow rule while the volumetric part is non-associated. For this case, $A_{ijkl}$ can be formulated as:

$$A_{ijkl} = \left( \beta \frac{1}{3} (\delta_{ij} \delta_{kl}) + \frac{1}{2} (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) \right)$$  \hspace{1cm} (104.12)

$$A_{ijkl}^{-1} = \left( -\beta + \frac{1}{1 + \beta} \frac{1}{3} (\delta_{ij} \delta_{kl}) + \frac{1}{2} (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) \right)$$  \hspace{1cm} (104.13)

and it is obvious that the non-associated flow rule is obtained with $\beta \neq 0$ and the associated flow rule with $\beta = 0$. It is useful to choose $\beta \geq 0$ and retain nice, positive definite properties of adjusted constitutive tensors later.

Let the $\| \cdot \|$ norm, be the complementary energy norm:

$$\|\sigma_{ij}\|^2 = \sigma_{ij} D_{ijkl} \sigma_{kl}$$  \hspace{1cm} (104.14)

where $D_{ijkl}$ is the elastic compliance tensor ($D_{ijkl} = E_{ijkl}^{-1}$), and let us introduce the adjusted complementary energy norm as:

$$A^\|\sigma_{ij}\|^2 = \sigma_{ij} (A_{ijkl} D_{klmn}) \sigma_{mn} = \sigma_{ij} (A D_{ijmn}) \sigma_{mn}$$  \hspace{1cm} (104.15)

where $A D_{ijmn}$ is the elastic compliance tensor transformed with respect to the non-associativity involved. It is clear that when $A_{ijkl} \equiv I_{ijkl}^{sym} \Rightarrow A^\|\sigma_{ij}\|^2 \equiv \|\sigma_{ij}\|^2$

### 104.2.2 On Integration Algorithms

In the section Constitutive Relations for Infinitesimal Plasticity we have summarized constitutive equations that are capable of representing a wide variety of elasto-plastic materials. The problem in Computational Elasto-plasticity is to devise accurate and efficient algorithms for the integration of such constitutive relations. In the context of finite element analysis using isoparametric elements, the integration of constitutive equations is carried out at Gauss points. In each step the deformation increments are given or known, and the unknowns to be found are updated stresses and plastic variables. According to Ortiz and Popov (1985) an acceptable algorithm should satisfy:

- consistency with the constitutive relations to be integrated or first order accuracy,
- Numerical stability,

---

8 This norm will be reintroduced later on!
• incremental plastic consistency

A non-required but desirable feature to be added to the above list is:

• higher\textsuperscript{9} order accuracy

First two conditions are needed for attaining convergence for the numerical solution as the step or increment becomes vanishingly small. The third condition is the algorithmic counterpart of the plastic consistency condition and requires that the state of stress computed from the algorithm be contained within the elastic domain.

### 104.2.3 Midpoint Rule Algorithm

A class of algorithms for integrating constitutive equations with potential to satisfy the above mentioned conditions are the Generalized Midpoint rule algorithms. They are given in the following form:

\[
d^{(n+1)\sigma_{ij}} = E_{ijkl} \left( d^{(n+1)\epsilon_{kl}} - d^{(n+1)p\epsilon_{kl}} \right) \tag{104.16}
\]

\[
d^{(n+1)p\epsilon_{ij}} = d^{(n)p\epsilon_{ij}} + d\lambda^{n+\alpha}m_{ij} \tag{104.17}
\]

\[
d^{(n+1)q_*} = d^{(n)q_*} + d\lambda^{n+\alpha}h_* \tag{104.18}
\]

\[F_{n+1} = 0 \tag{104.19}\]

where:

\[
n+\alpha m_{ij} = m_{ij} \left( (1 - \alpha) n\sigma_{ij} + \alpha \left( (n+1)\sigma_{ij} \right), (1 - \alpha) nq_* + \alpha \left( (n+1)q_* \right) \right) \tag{104.20}
\]

\[
n+\alpha h_* = h_* \left( (1 - \alpha) n\sigma_{ij} + \alpha \left( (n+1)\sigma_{ij} \right), (1 - \alpha) nq_* + \alpha \left( (n+1)q_* \right) \right) \tag{104.21}
\]

It is quite clear that the case $\alpha = 0$ corresponds to the Forward Euler approach\textsuperscript{10}, the case $\alpha = 1$ corresponds to the Backward Euler approach\textsuperscript{11}, and the case $\alpha = 1/2$ to the Crank–Nicholson scheme. Equations (104.16), (104.17), (104.18), (104.19), (104.20) and (104.21) are the nonlinear algebraic
Figure 104.1: integration algorithms in elasto–plasticity

equations to be solved for the unknowns $d(n+1\sigma_{ij})$, $d(n+1\epsilon_{ij}^p)$, $d(n+1q^*)$ and $d\lambda$. From the Figure (104.1)$^{12}$ it can be seen that the Generalized Midpoint rule may be regarded as a returning mapping algorithm in which the elastic predictor $\sigma_{ij}^{\text{pred}}$ is projected on the updated yield surface along the flow direction evaluated at the midpoint $(n+\alpha\sigma_{ij}, n+\alpha q^*)$.

104.2.3.1 Accuracy Analysis

Bearing in mind the context of the displacement based finite element analysis the integration of constitutive equations is performed for the given strain increment. The updated strains $n+1\epsilon_{ij} = \epsilon_{ij}(tn + \Delta t)$ may be viewed as the known function of the step size $\Delta t$. The remaining updated variables $n+1\sigma_{ij}$, $n+1\epsilon_{ij}^p$, $n+1q^*$, as well as the incremental plastic parameter $\lambda$ become functions of $\Delta t$ implicitly defined through equations (104.16), (104.17), (104.18) and (104.19). It should be clear from (104.16), (104.17), (104.18) and (104.19) that as $\Delta t \to 0$ than $n+1\epsilon_{ij} \to n\epsilon_{ij}$, and thus the limiting values of $n+1\sigma_{ij}$, $n+1\epsilon_{ij}^p$, $n+1q^*$ and $\lambda$ are obtained:

9at least second order accuracy.
10explicit scheme.
11implicit scheme.
12it should be pointed out that the vectors, as drawn on this figure, are pointing in the right direction only if we assume that $E_{ijkl} \equiv I_{ijkl}$. For any general elasticity tensor $E_{ijkl}$ all vectors are defined in the $E_{ijkl}$ metric, so the term ”normal”, as we are used to it, does not apply here.
\[
\lim_{\Delta t \to 0} (n^{+1}\sigma_{ij}) = n\sigma_{ij} \\
\lim_{\Delta t \to 0} (n^{+1}\epsilon_{ij}) = n\epsilon_{ij} \\
\lim_{\Delta t \to 0} (n^{+1}\epsilon_{ij}^p) = n\epsilon_{ij}^p \\
\lim_{\Delta t \to 0} (n^{+1}q_*) = nq_* \\
\lim_{\Delta t \to 0} \lambda = 0
\] (104.22)

It can also be argued that, by virtue of the implicit function theorem ((Abraham et al., 1988) Chapter 2.5), \(n^{+1}\sigma_{ij}, n^{+1}\epsilon_{ij}^p, n^{+1}q_*\) and \(\lambda\) are differentiable functions of \(\Delta t\), if the functions \(n^{+\alpha}m_{ij}, n^{+\alpha}h_*\) and \(F\) are sufficiently smooth. Sufficient smoothness will be assumed as needed.

**First Order Accuracy.** First order accuracy\(^{13}\) of the algorithm, defined by the equations (104.16), (104.17), (104.18) and (104.19) with the constitutive equations given by (104.1), (104.2), (104.3) and (104.4) necessitates that the numerically integrated variables \(n^{+1}\sigma_{ij}, n^{+1}\epsilon_{ij}^p\) and \(n^{+1}q_*\) agree with their exact values \(\sigma_{ij}(t + \Delta t), \epsilon_{ij}^p(t + \Delta t)\) and \(q_*(t + \Delta t)\) to within second order terms in the Taylor’s expansion around the initial state \(n\sigma_{ij} = \sigma_{ij}(t), n\epsilon_{ij}^p = \epsilon_{ij}^p(t)\) and \(nq_* = q_*(t)\) in \(\Delta t\). First order accuracy can be written in the following form:

\[
\lim_{\Delta t \to 0} \frac{d(n^{+1}\sigma_{ij})}{d(\Delta t)} = \frac{d(n\sigma_{ij})}{d(\Delta t)} = E_{ijkl} \left( \frac{d(n\epsilon_{ij})}{d(\Delta t)} - \frac{d(n\epsilon_{ij}^p)}{d(\Delta t)} \right) \\
\lim_{\Delta t \to 0} \frac{d(n^{+1}\epsilon_{ij}^p)}{d(\Delta t)} = \frac{d(n\epsilon_{ij}^p)}{d(\Delta t)} = \frac{d(n\epsilon_{ij})}{d(\Delta t)} n_{m_{ij}} \\
\lim_{\Delta t \to 0} \frac{d(n^{+1}q_*)}{d(\Delta t)} = \frac{d(nq_*)}{d(\Delta t)} = \frac{d(n\lambda)}{d(\Delta t)} n_{h_*} \\
\lim_{\Delta t \to 0} \frac{d(\lambda)}{d(\Delta t)} = \frac{d(n\lambda)}{d(\Delta t)} \] (104.23)

\(\lim_{\Delta t \to 0} \frac{d(\lambda)}{d(\Delta t)} = \frac{d(n\lambda)}{d(\Delta t)} \] (104.25)

\(\lim_{\Delta t \to 0} \frac{d(\lambda)}{d(\Delta t)} = \frac{d(n\lambda)}{d(\Delta t)} \] (104.26)

\(^{13}\)first order consistency.
and the plastic parameter \( d^{(n)} \lambda / d (\Delta t) \) is determined with the aid of the plastic consistency condition at \( t \):

\[
\frac{d^{(n)}F}{d (\Delta t)} = \frac{\partial (nF)}{\partial \sigma_{ij}} \frac{d \sigma_{ij}}{d (\Delta t)} + \frac{\partial (nF)}{\partial q_*} \frac{d q_*}{d (\Delta t)} = \eta_{nij} \frac{d \sigma_{ij}}{d (\Delta t)} + \eta_{n} \frac{d q_*}{d (\Delta t)} = 0
\]  

(104.27)

It is now rather straightforward to check whether the Generalized Midpoint rule satisfies the consistency conditions as given by (104.23), (104.24), (104.25) and (104.26). We can proceed further on by differentiating (104.16), (104.17), (104.18) and (104.19) with respect to \( \Delta t \):

\[
\frac{d^{(n+1)}\sigma_{ij}}{d (\Delta t)} = E_{ijkl} \left( \frac{d^{(n+1)}\epsilon_{kl}}{d (\Delta t)} - \frac{d^{(n+1)}\epsilon_{kl}^p}{d (\Delta t)} \right)
\]  

(104.28)

\[
\frac{d^{(n+1)}q_*}{d (\Delta t)} = \frac{d \lambda}{d (\Delta t)} (n^{+\alpha}m_{ij}) + \lambda \left( \frac{\partial m_{ij}}{\partial \sigma_{ij}} \bigg|_{n+1} \frac{d^{(n+1)}\sigma_{ij}}{d (\Delta t)} + \frac{\partial m_{ij}}{\partial q_*} \bigg|_{n+1} \frac{d^{(n+1)}q_*}{d (\Delta t)} \right)
\]  

(104.29)

\[
\frac{d^{(n+1)}h_*}{d (\Delta t)} = \frac{d \lambda}{d (\Delta t)} (n^{+\alpha}h_*) + \lambda \left( \frac{\partial h_*}{\partial \sigma_{ij}} \bigg|_{n+1} \frac{d^{(n+1)}\sigma_{ij}}{d (\Delta t)} + \frac{\partial h_*}{\partial q_*} \bigg|_{n+1} \frac{d^{(n+1)}q_*}{d (\Delta t)} \right)
\]  

(104.30)

\[
\frac{d^{(n+1)}F}{d (\Delta t)} = \frac{\partial (n^{+1}F)}{\partial (n^{+1}\sigma_{ij})} \frac{d^{(n+1)}\sigma_{ij}}{d (\Delta t)} + \frac{\partial (n^{+1}F)}{\partial (n^{+1}q_*)} \frac{d^{(n+1)}q_*}{d (\Delta t)} = 0
\]  

(104.31)

where \( n^{+\alpha}m_{ij} \) and \( n^{+\alpha}h_\alpha \) are defined by the equations (104.20) and (104.21).

By taking \( \Delta t \) to the limit value, \( \Delta t \rightarrow 0 \), in the (104.28), (104.29), (104.30) and (104.31) and using the relations from (104.22) one finds:

\[
\lim_{\Delta t \to 0} \frac{d^{(n+1)}\sigma_{ij}}{d (\Delta t)} = E_{ijkl} \left( \frac{d^{(n)}\epsilon_{kl}}{d (\Delta t)} - \frac{d^{(n+1)}\epsilon_{kl}^p}{d (\Delta t)} \right)
\]  

(104.32)

\footnote{bearing in mind that values at \( t \) are constants and that only variables at \( t + \Delta t \) are changing with respect to \( \Delta t \).}
\[
\lim_{\Delta t \to 0} \frac{d \left( n^{+1} \epsilon_{ij} \right)}{d (\Delta t)} = \frac{d \lambda}{d (\Delta t)} (n^{i} m_{ij})
\]

(104.33)

\[
\lim_{\Delta t \to 0} \frac{d \left( n^{+1} q_{*} \right)}{d (\Delta t)} = \frac{d \lambda}{d (\Delta t)} (n^{h}_{*})
\]

(104.34)

\[
\lim_{\Delta t \to 0} \frac{d (n^{+1} F)}{d (\Delta t)} = \frac{\partial (nF)}{\partial \sigma_{ij}} \left( \lim_{\Delta t \to 0} \frac{d \left( n^{+1} \sigma_{ij} \right)}{d (\Delta t)} \right) + \frac{\partial (nF)}{\partial q_{*}} \left( \lim_{\Delta t \to 0} \frac{d \left( n^{+1} q_{*} \right)}{d (\Delta t)} \right) = 0
\]

(104.35)

In the previous equations it is quite clear that since \( \Delta t = 0 \), then equations (104.22) hold and since the variables \( n^{\sigma_{ij}}, n^{\epsilon_{ij}^{p}} \) and \( n^{q_{*}} \) are constant with respect to the change in \( \Delta t \), the result follows readily, i.e. the Midpoint rule satisfies first order accuracy.

**Second Order Accuracy** To investigate second order accuracy of the algorithm given by (104.16), (104.17), (104.18) and (104.19) together with the constitutive equations given by (104.1), (104.2), (104.3) and (104.4) we shall proceed in the following manner. Second order accuracy actually means that the numerically integrated variables \( n^{+1} \sigma_{ij}, n^{+1} \epsilon_{ij}^{p} \) and \( n^{+1} q_{*} \) agree with their "exact" values \( \sigma_{ij}(t + \Delta t), \epsilon_{ij}^{p}(t + \Delta t) \) and \( q_{*}(t + \Delta t) \) to within third order terms in the Taylor’s expansion around the initial state \( n^{\sigma_{ij}} = \sigma_{ij}(t), n^{\epsilon_{ij}^{p}} = \epsilon_{ij}^{p}(t) \) and \( n^{q_{*}} = q_{*}(t) \) in \( \Delta t \). This verbal statement can be written in the following mathematical form:

\[
E_{ijkl} \left( \lim_{\Delta t \to 0} \frac{d^{2} \left( n^{+1} \epsilon_{kl} \right)}{d (\Delta t)^{2}} - \lim_{\Delta t \to 0} \frac{d^{2} \left( n^{p} \epsilon_{kl} \right)}{d (\Delta t)^{2}} \right) = E_{ijkl} \left( \frac{d^{2} \left( n^{p} \epsilon_{kl} \right)}{d (\Delta t)^{2}} - \frac{d^{2} \left( n^{p} \epsilon_{kl} \right)}{d (\Delta t)^{2}} \right) = 0
\]

(104.36)
\[
\lim_{\Delta t \to 0} \frac{d^2 (n+1)p_{ij}}{d(\Delta t)^2} =
\]

\[
\frac{d^2 \lambda}{d(\Delta t)^2} \left( \frac{d^{(n+1)}m_{ij}}{d(\Delta t)} + \frac{d^{(n+1)}\lambda}{d(\Delta t)} \frac{d^{(n+1)}m_{ij}}{d(\Delta t)} \right) =
\]

\[
\frac{d^2 \lambda}{d(\Delta t)^2} \left( \frac{d^{(n)}m_{ij}}{d(\Delta t)} + \frac{d^{(n)}\lambda}{d(\Delta t)} \frac{d^{(n)}m_{ij}}{d(\Delta t)} \right) =
\]

\[
\frac{d^2 \lambda}{d(\Delta t)^2} \left( \frac{d^{(n)}m_{ij}}{d(\Delta t)} + \frac{d^{(n)}\lambda}{d(\Delta t)} \frac{\partial m_{ij}}{\partial \sigma_{ij}} \bigg|_n \frac{d^{(n)}\sigma_{ij}}{d(\Delta t)} + \frac{\partial m_{ij}}{\partial q_\sigma} \bigg|_n \frac{d^{(n)}q_\sigma}{d(\Delta t)} \right)
\]

(104.37)

\[
\lim_{\Delta t \to 0} \frac{d^2 (n+1)q_\sigma}{d(\Delta t)^2} =
\]

\[
\frac{d^2 \lambda}{d(\Delta t)^2} \left( \frac{d^{(n+1)}h_\sigma}{d(\Delta t)} + \frac{d^{(n+1)}\lambda}{d(\Delta t)} \frac{d^{(n+1)}h_\sigma}{d(\Delta t)} \right) =
\]

\[
\frac{d^2 \lambda}{d(\Delta t)^2} \left( \frac{d^{(n+1)}h_\sigma}{d(\Delta t)} + \frac{d^{(n+1)}\lambda}{d(\Delta t)} \frac{\partial h_\sigma}{\partial \sigma_{ij}} \bigg|_n \frac{d^{(n)}\sigma_{ij}}{d(\Delta t)} + \frac{\partial h_\sigma}{\partial q_\sigma} \bigg|_n \frac{d^{(n)}q_\sigma}{d(\Delta t)} \right)
\]

(104.38)

\[
\lim_{\Delta t \to 0} \frac{d^2 (\lambda)}{d(\Delta t)^2} = \frac{d^2 (n)\lambda}{d(\Delta t)^2}
\]

(104.39)

and the plastic parameter \(\frac{d^2 (n)\lambda}{d(\Delta t)^2}\) is determined with the aid of the second order oscillatory satisfaction of the plastic consistency condition:

\[
\frac{d^2 (n)F}{d(\Delta t)^2} = \frac{dn_{ij}}{d\Delta t} \bigg|_n \frac{d\sigma_{ij}}{d(\Delta t)} + n_{ij} \frac{d^2 (n)\sigma_{ij}}{d(\Delta t)^2} + \frac{d(\xi_\sigma)}{d(\Delta t)} \bigg|_n \frac{d^{(n)}q_\sigma}{d(\Delta t)} + n_{\xi_\sigma} \frac{d^2 (n)q_\sigma}{d(\Delta t)^2} = 0
\]

(104.40)

Now we can proceed by taking the second derivative of the equations (104.16), (104.17), (104.18) and (104.19) or use the already derived first derivatives from equations (104.28), (104.29), (104.30) and (104.31), and then differentiate them again so that we get:

\[
\frac{d^2 (n+1)\epsilon_{ij}}{d(\Delta t)^2} = E_{ijkl} \left( \frac{d^2 (n+1)\epsilon_{kl}}{d(\Delta t)^2} - \frac{d^2 (n+1)p_{kl}}{d(\Delta t)^2} \right)
\]

(104.41)
\[
\frac{d^2 (n+1 \epsilon_{ij}^p)}{d (\Delta t)^2} = \\
\frac{d^2 \lambda (n+\alpha m_{ij})}{d (\Delta t)^2} + \\
2 \frac{d \lambda}{d (\Delta t)\alpha} \left( \frac{\partial m_{ij}}{\partial \sigma_{ij}} \bigg|_{n+1} \frac{d (n+1 \sigma_{ij})}{d (\Delta t)} + \frac{\partial m_{ij}}{\partial q_{*}} \bigg|_{n+1} \frac{d (n+1 q_{*})}{d (\Delta t)} \right) + \\
\lambda \alpha \frac{d}{d (\Delta t)} \left( \frac{\partial m_{ij}}{\partial \sigma_{ij}} \bigg|_{n+1} \frac{d (n+1 \sigma_{ij})}{d (\Delta t)} + \frac{\partial m_{ij}}{\partial q_{*}} \bigg|_{n+1} \frac{d (n+1 q_{*})}{d (\Delta t)} \right) 
\] (104.42)

\[
\frac{d^2 (n+1 \epsilon_{kl}^e)}{d (\Delta t)^2} = \\
\frac{d^2 \lambda (n+\alpha h_{*})}{d (\Delta t)^2} + \\
2 \frac{d \lambda}{d (\Delta t)\alpha} \left( \frac{\partial h_{*}}{\partial \sigma_{ij}} \bigg|_{n+1} \frac{d (n+1 \sigma_{ij})}{d (\Delta t)} + \frac{\partial h_{*}}{\partial q_{*}} \bigg|_{n+1} \frac{d (n+1 q_{*})}{d (\Delta t)} \right) + \\
\lambda \alpha \frac{d}{d (\Delta t)} \left( \frac{\partial h_{*}}{\partial \sigma_{ij}} \bigg|_{n+1} \frac{d (n+1 \sigma_{ij})}{d (\Delta t)} + \frac{\partial h_{*}}{\partial q_{*}} \bigg|_{n+1} \frac{d (n+1 q_{*})}{d (\Delta t)} \right) 
\] (104.43)

\[
\frac{d^2 (n+1 F)}{d (\Delta t)^2} = \\
\frac{d (n+1 \epsilon_{ij}^e)}{d \sigma_{ij}} \frac{d (n+1 \sigma_{ij})}{d (\Delta t)} + n+1 \epsilon_{ij} \frac{d^2 (n+1 \sigma_{ij})}{d (\Delta t)^2} + \\
\frac{d (n+1 \xi_{*})}{d \sigma_{ij}} \frac{d (n+1 q_{*})}{d (\Delta t)} + n+1 \xi \frac{d^2 (n+1 q_{*})}{d (\Delta t)^2} = 0 
\] (104.44)

If we drive \(\Delta t\) to the limit, namely by taking \(\lim_{\Delta t \to 0}\) and keeping in mind equations (104.22) and the assumed consistency of the algorithm\(^{15}\) as given by the equations (104.23), (104.24), (104.25) and (104.26) one finds:

\[
\lim_{\Delta t \to 0} \frac{d^2 (n+1 \sigma_{ij})}{d (\Delta t)^2} = E_{ijkl} \left( \frac{d^2 (\epsilon_{kl}^e)}{d (\Delta t)^2} - \lim_{\Delta t \to 0} \frac{d^2 (n+1 \epsilon_{kl}^p)}{d (\Delta t)^2} \right) 
\] (104.45)

\(^{15}\)actually the first order accuracy that is already proven.
\[
\lim_{\Delta t \to 0} \frac{d^2 \left( n^1 \lambda \right)}{d (\Delta t)^2} (n^{+} \alpha_{ij}) + 2 \frac{d \left( n \lambda \right)}{d (\Delta t)} \alpha \left( \frac{\partial m_{ij}}{\partial \sigma_{ij}} \bigg|_{n} \frac{d \left( n \sigma_{ij} \right)}{d (\Delta t)} + \frac{\partial m_{ij}}{\partial q_{*}} \bigg|_{n} \frac{d \left( n q_{*} \right)}{d (\Delta t)} \right) = \]

(104.46)

\[
\lim_{\Delta t \to 0} \frac{d^2 \left( n^1 q_{*} \right)}{d (\Delta t)^2} =
\]

(104.47)

\[
\lim_{\Delta t \to 0} \frac{d^2 \left( n^1 F \right)}{d (\Delta t)^2} =
\]

(104.48)

By comparing equations (104.45), (104.46), (104.47) and (104.48) with the second order accuracy condition stated in equations (104.36), (104.37), (104.38) and (104.39) it is quite clear that the second order accuracy is obtained if\(^{16}\) \( \alpha = 1/2 \) !

The conclusion is that the Midpoint–rule algorithm is consistent\(^{17}\) for all \( \alpha \in [0, 1] \) and it is second order accurate for \( \alpha = 1/2 \). However, one should not forget that these results are obtained for the limiting case \( \Delta t \to 0 \), i.e. the strain increments are small and tend to zero.

### 104.2.3.2 Numerical Stability Analysis

Numerical stability of an algorithm plays a central role in approximation theory for initial value problems. In fact, it can be stated that consistency and stability are necessary and sufficient conditions for convergence of an algorithm as the time step tends to zero. In the approach presented by Ortiz and Popov\(^{16}\) if and only if \( (\iff) \).

\(^{17}\)first order accurate.
A new methodology is proposed by which the stability properties of an integration algorithm for elasto–plastic constitutive relations can be established. Our attention is confined to perfect plasticity and a smooth yield surface.

The purpose of the following stability analysis is to determine under what conditions a finite perturbation in the initial stresses is diluted by the algorithm. In other words:

\[
d\left( n+1 \sigma_{ij}^{(2)}, n+1 \sigma_{ij}^{(1)} \right) \leq d\left( n \sigma_{ij}^{(2)}, n \sigma_{ij}^{(1)} \right)
\]

where \( d(\cdot, \cdot) \) is some suitable distance on the yield surface and \( n+1 \sigma_{ij}^{(1)} \) and \( n+1 \sigma_{ij}^{(2)} \) are two sets of updated stresses corresponding to arbitrary initial stress values \( n \sigma_{ij}^{(1)} \) and \( n \sigma_{ij}^{(2)} \), respectively, and all of the previous stress values are assumed to lie on the yield surface. Stability in the sense of equation (104.49) is referred to as large scale stability. It is shown in Helgason (1978)\(^\text{18}\) that for nonlinear initial value problems defined on Banach manifolds, consistency and large scale stability with respect to a complete metric are sufficient for convergence.

The task of directly establishing estimates of the type expressed in (104.49) is rather difficult, and so despite the conceptual appeal of large scale stability, simplified solutions are sought. It should be recognized that attention can be restricted to infinitesimal perturbation in the initial conditions of the type \( n \sigma_{ij} \rightarrow n \sigma_{ij} + d(n \sigma_{ij}) \). This simplification is founded on the fact that the dilution or attenuation, by the algorithm of infinitesimal perturbations:

\[
\| d\ n+1 \sigma_{ij} \| \leq \| d\ n \sigma_{ij} \|
\]

with respect to some suitable norm \( \| \cdot \| \), of small scale stability, implies large scale stability in the sense of equation (104.49).

Let the \( \| \cdot \| \) norm, be the energy norm:

\[
\| \sigma_{ij} \|^2 = \sigma_{ij} D_{ijkl} \sigma_{kl}
\]

where \( D_{ijkl} \) is the elastic compliance tensor (\( D_{ijkl} = E_{ijkl}^{-1} \)), and let the distance \( d(\cdot, \cdot) \) on the yield surface be defined as

\[
d\left( \sigma_{ij}^{(1)}, \sigma_{ij}^{(2)} \right) = \inf_{\gamma} \int_{\gamma} \| \sigma'_{ij}(s) \| ds
\]

\(^{18}\)the first Chapter of Helgason’s book.
where the infimum is taken over all possible stress paths \( \gamma \) on the yield surface that are joining two stress states, namely \( \sigma_{ij}^{(1)} \) and \( \sigma_{ij}^{(2)} \). It can be found in Helgason (1978) that for a smooth yield surface, equation (104.52) defines the geodesic distance which endows the yield surface with a complete metric structure.

Suppose that we have any two initial states of stress \( \sigma_{ij}^{(1)} \) and \( \sigma_{ij}^{(2)} \) and let \( \sigma_{ij}^{(1)+1} \) and \( \sigma_{ij}^{(2)+1} \) be the corresponding updated values, respectively, and all the previous stress states are assumed to lie on the yield surface. Then, according to Helgason (1978), there exists a unique geodesic curve that joins \( \sigma_{ij}^{(1)} \) and \( \sigma_{ij}^{(2)} \) for which the infimum in equation (104.52) is attained. If \( \gamma_n \) is such a curve, then by definition:

\[
d(\sigma_{ij}^{(1)}, \sigma_{ij}^{(2)}) = \int_{\gamma_n} \| \sigma_{ij}'(s) \| ds \quad (104.53)
\]

Let the new curve \( \gamma_{n+1} \) be the transform of curve \( \gamma_n \) by the algorithm. By definition \( \gamma_{n+1} \) lies on the yield surface and joins two stress states \( \sigma_{ij}^{(1)+1} \) and \( \sigma_{ij}^{(2)+1} \). By the definition given in (104.52), it follows that:

\[
d(\sigma_{ij}^{(1)+1}, \sigma_{ij}^{(2)+1}) = \int_{\gamma_{n+1}} \| \sigma_{ij}'(s) \| ds \quad (104.54)
\]

Under the assumption of small scale stability of the algorithm one can write:

\[
\| \sigma_{ij}'(s_{n+1}) \| ds = \| d\sigma_{ij}(s_{n+1}) \| \leq \| d\sigma_{ij}(s_n) \| = \| \sigma_{ij}'(s_n) \| ds \quad (104.55)
\]

for every pair of corresponding points \( s_n \) and \( s_{n+1} \) on \( \gamma_n \) and \( \gamma_{n+1} \) respectively, so it follows:

\[
\int_{\gamma_{n+1}} \| \sigma_{ij}'(s_{n+1}) \| ds \leq \int_{\gamma_n} \| \sigma_{ij}'(s_n) \| ds \quad (104.56)
\]

By combining equations (104.54), (104.55) and (104.56) it is concluded that:

\[
d(\sigma_{ij}^{(1)+1}, \sigma_{ij}^{(2)+1}) \leq d(\sigma_{ij}^{(1)}, \sigma_{ij}^{(2)}) \quad (104.57)
\]

which proves large scale stability. The main conclusion of the above argument may be stated as follows: small scale stability in the energy norm is equivalent to large scale stability in the associated geodesic distance.
The previous result is of practical importance, since it shows that the stability analysis for the integration algorithm in elasto–plasticity can be carried out by the assessment of small scale stability. The small scale stability analysis of the Generalized Midpoint rule is necessary to determine how the algorithm propagates infinitesimal perturbations in the initial conditions. By differentiating equations (104.16), (104.17), (104.18) and (104.19) and considering that we are dealing with perfectly plastic case here so that \((n+1)q_\ast = (n)q_\ast\) = constants, it follows:

\[
d (n+1)\sigma_{ij} = -E_{ijkl} d (n+1)p_{kl}
\]

(104.58)

\[
d (n)\sigma_{ij} = -E_{ijkl} d (n)p_{kl}
\]

(104.59)

\[
d (n+1)\epsilon_{ij}^p - d (n)p_{ij} = d \lambda \ (n+\alpha)m_{ij} + \lambda \ d (n+\alpha)m_{ij}
\]

(104.60)

\[
d (n+1)F = \frac{\partial F}{\partial \sigma_{ij}} \bigg|_{n+1} d (n+1)\sigma_{ij}) = n+1n_{ij} \ d (n+1)\sigma_{ij} = 0
\]

(104.61)

Let us now examine the shape of \(d (n+\alpha)m_{ij}\) having in mind the original definition given in equation (104.20):

\[
n+\alpha m_{ij} = m_{ij} \ (1 - \alpha) \ n\sigma_{ij} + \alpha \ (n+1)\sigma_{ij} + \ (1 - \alpha) \ nq_\ast + \alpha \ (n+1)q_\ast.
\]

and the differential of the previous equation is:

\[
d (n+\alpha)m_{ij} = (1 - \alpha) \ \frac{\partial m_{ij}}{\partial \sigma_{kl}} \bigg|_{n+\alpha} d (n)\sigma_{kl} + \alpha \ \frac{\partial m_{ij}}{\partial \sigma_{kl}} \bigg|_{n+\alpha} d (n+1)\sigma_{kl}
\]

To ease writing let us introduce the following fourth order tensor:

\[
19 the remark about restraining analysis to perfectly plastic case still holds, so that \((n+1)q_\ast\) and \((n)q_\ast\) are constant.
\( M_{ijkl} = \frac{\partial m_{ij}}{\partial \sigma_{kl}} \)

The equation (104.60) now reads:

\[
d\left( n + 1 \epsilon_{ij}^p \right) - d\left( n \epsilon_{ij}^p \right) = \\
d\lambda \left( n + \alpha \epsilon_{ij}^p \right) + \lambda \left( 1 - \alpha \right) \left( n + \alpha M_{ijkl} \right) d\left( n \sigma_{kl} \right) + \alpha \left( n + \alpha M_{ijkl} \right) d\left( n + 1 \sigma_{kl} \right)
\]

(104.62)

By using equations (104.58) and (104.59) and knowing that \( E_{ijkl}^{-1} = D_{ijkl} \) one can write:

\[
d\left( n + 1 \epsilon_{ij}^p \right) = - D_{ijkl} d\left( n + 1 \sigma_{kl} \right)
\]

\[
d\left( n \epsilon_{ij}^p \right) = - D_{ijkl} d\left( n \sigma_{kl} \right)
\]

so that the equation (104.62) now reads:

\[
-D_{ijkl} d\left( n + 1 \sigma_{kl} \right) + D_{ijkl} d\left( n \sigma_{kl} \right) = \\
d\lambda \left( n + \alpha \epsilon_{ij}^p \right) + \lambda \left( 1 - \alpha \right) \left( n + \alpha M_{ijkl} \right) d\left( n \sigma_{kl} \right) + \alpha \left( n + \alpha M_{ijkl} \right) d\left( n + 1 \sigma_{kl} \right)
\]

Now we are proceeding by solving the previous equation for \( d\left( n + 1 \sigma_{kl} \right) \):

\[
(D_{ijkl} + \lambda \alpha \left( n + \alpha M_{ijkl} \right)) d\left( n + 1 \sigma_{kl} \right) = \\
(D_{ijkl} - \lambda \left( 1 - \alpha \right) \left( n + \alpha M_{ijkl} \right)) d\left( n \sigma_{kl} \right) - d\lambda \left( n + \alpha \epsilon_{ij}^p \right)
\]
and by denoting:

\[ \Psi_{ijkl} = D_{ijkl} - \lambda (1 - \alpha) (n^{+\alpha} M_{ijkl}) \]

\[ \Gamma_{ijkl} = D_{ijkl} + \lambda \alpha (n^{+\alpha} M_{ijkl}) \]

it follows:

\[ d (n^{+1}\sigma_{kl}) = \Gamma_{ijkl}^{-1} \Psi_{ijkl} d (n\sigma_{kl}) - d\lambda (n^{+\alpha} m_{ij}) \] (104.63)

Then by inserting the solution for \( d (n^{+1}\sigma_{kl}) \) in the consistency condition (104.61):

\[ d (n^{+1}F) = n^{+1} n_{kl} d (n^{+1}\sigma_{kl}) = 0 \]

one gets:

\[ d (n^{+1}F) = n^{+1} n_{kl}^{-1} \Psi_{ijkl} d (n\sigma_{kl}) - d\lambda (n^{+\alpha} m_{ij}) = 0 \] (104.64)

then if we solve for \( d\lambda \):

\[ d\lambda n^{+\alpha} m_{ij} n^{+1} n_{kl}^{-1} \Gamma_{ijkl}^{-1} = n^{+1} n_{kl}^{-1} \Gamma_{ijkl}^{-1} \Psi_{ijkl} d (n\sigma_{kl}) \] (104.65)

or\(^{20}\):

\(^{20}\) where the change in dummy indices is possible because \( d\lambda \) is scalar.
\[
\frac{d\lambda}{d} = \frac{n+1 r_{rs} \Gamma^{-1}_{pqrs} \Psi_{pqrs} d (\sigma_{rs})}{(n+\alpha_m_{pq})(n+1 r_{rs}) \Gamma^{-1}_{pqrs}} \tag{104.66}
\]

then by using the solution for \(d (n+1 \sigma_{kl})\) from (104.63) and the solution for \(d\lambda\) from (104.66) one can find:

\[
d (n+1 \sigma_{kl}) = \Gamma^{-1}_{ijkl} \Psi_{ijkl} d (\sigma_{kl}) - \Gamma^{-1}_{pqrs} \Psi_{pqrs} \frac{n+1 r_{rs} \Gamma^{-1}_{ijkl} (n+\alpha_m_{ij})}{n+\alpha_m_{pq} n+1 r_{rs} \Gamma^{-1}_{pqrs}} d (\sigma_{rs}) \tag{104.67}
\]

\[
d (n+1 \sigma_{kl}) = \Gamma^{-1}_{ijkl} \Psi_{ijkl} \left(\delta_{ks} \delta_{rl} - \frac{n+1 r_{rs} \Gamma^{-1}_{ijkl} (n+\alpha_m_{ij})}{n+\alpha_m_{pq} n+1 r_{rs} \Gamma^{-1}_{pqrs}}\right) d (\sigma_{rs}) \tag{104.68}
\]

to ease the writing we can define the following notation:

\[
\Phi_{klrs} = \delta_{ks} \delta_{rl} - \frac{n+1 r_{rs} \Gamma^{-1}_{ijkl} (n+\alpha_m_{ij})}{n+\alpha_m_{pq} n+1 r_{rs} \Gamma^{-1}_{pqrs}} \tag{104.69}
\]

so that the equation (104.68) now reads:

\[
d (n+1 \sigma_{kl}) = \Gamma^{-1}_{ijkl} \Psi_{ijkl} \Phi_{klrs} d (\sigma_{rs}) \tag{104.70}
\]

In order to derive the estimate of the type (104.50) from (104.70) we shall proceed in the following way. The norm of a tensor is defined as:

\[
\|A_{ijkl}\| = \sup_{\sigma} \frac{\|A_{ijkl}\sigma_{kl}\|}{\|\sigma_{kl}\|} \tag{104.71}
\]

If we take the norm of (104.70), while recalling the inequalities:
\[ \|A_{ijkl}\| \leq \|A_{ijkl}\| \|\sigma_{kl}\| \quad ; \quad \|A_{ijkl}B_{ijkl}\| \leq \|A_{ijkl}\| \|B_{ijkl}\| \] (104.72)

it follows:

\[ \|d(n+1\sigma_{kl})\| = \|\Gamma^{-1}_{ijkl}\Psi_{ijkl}\Phi_{klrs}d(n\sigma_{rs})\| \] (104.73)

then by using equations (104.72), we are able to write:

\[ \|d(n+1\sigma_{kl})\| \leq \|\Gamma^{-1}_{ijkl}\Psi_{ijkl}\| \|\Phi_{klrs}\| \|d(n\sigma_{rs})\| \] (104.74)

Considering the norm of \(\|\Phi_{klrs}\|\) it should be noted that \(\Phi_{klrs}\) defines a projection along the direction of \(\Gamma^{-1}_{ijkl}n+\alpha_{mij}\) onto the hyperplane that is orthogonal to \(n+1n_{rs}\), so that the following properties hold:

\[ (\Phi_{klrs})\left(\Gamma^{-1}_{ijkl}n+\alpha_{mij}\right) = \emptyset \] (104.75)

\[ (\Phi_{klrs})(\sigma_{rs}) = \sigma_{rs} \] (104.76)

for every \(\sigma_{rs}\) that is orthogonal to \(n+1n_{rs}\). From these properties and the definition in equation (104.71) it follows that:

\[ \|\Phi_{klrs}\| \equiv 1 \] (104.77)

In what follows it is assumed that the fourth order tensor field
\( M_{ijkl} = \partial m_{ij} / \partial \sigma_{kl} \)

is symmetric and positive definite everywhere on the yield surface. The assumption is valid, if the flow direction \( m_{ij} \) is derived from the convex potential function, which is a rather common feature among yield criteria. It is now clear that:

\[
\| \Gamma_{ijkl}^{-1} \Psi_{ijkl} \| = \frac{\max_{ij} \Psi_{ijkl} \max_{kl} \gamma_{kl}}{\max_{ij} \Gamma_{ijkl} \max_{kl} \gamma_{kl}}
\]  \hspace{1cm} (104.78)

where \( \max_{ij} \) is the eigentensor corresponding to the maximum eigenvalue of the eigenproblem:

\[
(\Psi_{ijkl} - \mu \Gamma_{ijkl}) \gamma_{kl} = 0
\]  \hspace{1cm} (104.79)

which is normalized to satisfy:

\[
\| \max_{ij} \| \| \max_{ij} \| = \max_{ij} D_{ijkl} \max_{kl} \gamma_{kl} = 1
\]  \hspace{1cm} (104.80)

If we denote:

\[
n^{+}M_{ijkl} = \max_{ij} \max_{ij} M_{ijkl}
\]  \hspace{1cm} (104.81)

as the maximum eigenvalue of the fourth order tensor \( n^{+}M_{ijkl} \) and that value is a positive real number\(^{21}\), then from equations (104.78), (104.80), (104.81) and from the definition\(^{22}\) of \( \Psi_{ijkl} \) and \( \Gamma_{ijkl} \), it follows:

\[
\| \Gamma_{ijkl}^{-1} \Psi_{ijkl} \| = \left| \frac{1 - (1 - \alpha) \lambda \left( n^{+}M_{ijkl} \right)}{1 + \alpha \lambda \left( n^{+}M_{ijkl} \right)} \right|
\]  \hspace{1cm} (104.82)

\(^{21}\) because \( n^{+}M_{ijkl} \) is derived from a convex potential function.

\(^{22}\) \( \Psi_{ijkl} = D_{ijkl} - \lambda \left( 1 - \alpha \right) \left( n^{+}M_{ijkl} \right) \) and \( \Gamma_{ijkl} = D_{ijkl} + \lambda \alpha \left( n^{+}M_{ijkl} \right) \)
which, when inserted in the equation (104.74) yields:

\[
\| d^{(n+1)\sigma_{kl}} \| \leq \left| \frac{1 - (1 - \alpha) \lambda (n+\alpha \beta)}{1 + \alpha \lambda (n+\alpha \beta)} \right| \| d^{(n)\sigma_{rs}} \| \quad (104.83)
\]

Since it is said that \( n+\alpha \beta \) is a positive real number it follows that:

\[
\left| \frac{1 - (1 - \alpha) \lambda (n+\alpha \beta)}{1 + \alpha \lambda (n+\alpha \beta)} \right| \leq \left| \frac{1 - \alpha}{\alpha} \right| \frac{n+\alpha \beta}{n+\alpha \beta} = \frac{1 - \alpha}{\alpha} \quad (104.84)
\]

and \( \alpha \in [0, 1] \). The new form of equation (104.83) is now:

\[
\| d^{(n+1)\sigma_{kl}} \| \leq \left| \frac{1 - \alpha}{\alpha} \right| \| d^{(n)\sigma_{rs}} \| \quad (104.85)
\]

which in conjunction with the requirement for unconditional stability\(^{23}\) yields:

\[
\left| \frac{1 - \alpha}{\alpha} \right| \leq 1 \quad (104.86)
\]

and so it is necessary that:

\[
\alpha \geq \min \alpha = \frac{1}{2} \quad (104.87)
\]

The conclusion is that the Generalized Midpoint rule is unconditionally stable for \( \alpha \geq 1/2 \). In the case when \( \alpha < 1/2 \) the Generalized Midpoint rule is only conditionally stable. To obtain a stability condition for \( \alpha \leq 1/2 \) one has to go back to equation (104.83), and we conclude that:

\[
\left| \frac{1 - (1 - \alpha) \lambda (n+\alpha \beta)}{1 + \alpha \lambda (n+\alpha \beta)} \right| \leq 1 \quad \Rightarrow \quad \lambda \leq \frac{2}{\max \beta \left(1 - 2\alpha\right)} \quad \text{for} \quad \alpha \leq \frac{1}{2} \quad (104.88)
\]

and when \( \alpha = 1/2 \), then \( \text{critical} \lambda \rightarrow \infty \), and thus the unconditional stability is recovered.

\(^{23}\) that is \( \| d^{n+1}\sigma_{ij} \| \leq \| d^n\sigma_{ij} \| \)
104.2.4 Crossing the Yield Surface

Midpoint rule algorithms in computational elasto–plasticity require the evaluation of the intersection stress. Despite the appeal of the closed form solution, as found in Bičanić (1989), and numerical iterative procedures as found in Marques (1984) and Nayak and Zienkiewicz (1972), for some yield criteria the solution is not that simple to find. Special problems arises, even with the numerical iterative methods in the area of a apex. The apex area problems are connected to the derivatives of yield a function.

![Figure 104.2: The pictorial representation of the intersection point problem in computational elasto–plasticity: which must be resolved for the Forward and Midpoint schemes](image)

Having in mind the before mentioned problems, a different numerical scheme, that does not need derivatives, was sought for solving this problem. One possible solution was found in Press et al. (1988b)

---

24 except for the fully implicit Backward Euler algorithm.

25 contact, penetration point, i.e the point along the stress path where \( F = 0 \) or the point where stress state crosses from the elastic to the plastic region.

26 namely for the MRS–Lade elasto–plastic model.
in the form of an excellent algorithm that combines root bracketing, bisection, and inverse quadratic interpolation to converge from a neighborhood of a zero crossing. The algorithm was developed in the 1960s by van Wijngaarden, Dekker and others at the Mathematical Center in Amsterdam. The algorithm was later improved by Brent, and so it is better known as Brent’s method. The method is guaranteed to converge, so long as the function \(^{27}\) can be evaluated within the initial interval known to contain a root. While the other iterative methods that do not require derivatives\(^{28}\) assume approximately linear behavior between two prior estimates, inverse quadratic interpolation uses three prior points to fit an inverse quadratic function\(^{29}\), whose value at \(y = 0\) is taken as the next estimate of the root \(x\). Lagrange’s classical formula for interpolating the polynomial of degree \(N - 1\) through \(N\) points \(y_1 = f(x_1), y_2 = f(x_2), \ldots, y_3 = f(x_3)\) is given by:

\[
P(x) = \frac{(x - x_2)(x - x_3) \cdots (x - x_N)}{(x_1 - x_2)(x_1 - x_3) \cdots (x_1 - x_N)} y_1 + \frac{(x - x_1)(x - x_3) \cdots (x - x_N)}{(x_2 - x_1)(x_2 - x_3) \cdots (x_2 - x_N)} y_2 + \cdots + \frac{(x - x_1)(x - x_2) \cdots (x - x_N)}{(x_N - x_1)(x_N - x_3) \cdots (x_N - x_{N-1})} y_N
\]

(104.89)

If the three point pairs are \([a, f(a)], [b, f(b)], [c, f(c)]\), then the interpolating formula (104.89) yields:

\[
x = \frac{(y - f(a))(y - f(b))}{(f(c) - f(a))(f(c) - f(b))} c + \frac{(y - f(b))(y - f(c))}{(f(a) - f(b))(f(a) - f(c))} a + \frac{(y - f(c))(y - f(a))}{(f(b) - f(a))(f(b) - f(a))} b
\]

(104.90)

By setting \(y = 0\), we obtain a result for the next root estimate, which can be written as:

\[
x = b + \frac{f(b)}{f(a)} \left( \frac{f(a)}{f(c)} - \frac{f(b)}{f(c)} \right) \left( c - b \right) - \left( 1 - \frac{f(b)}{f(c)} \right) \left( b - a \right)
\]

(104.91)

\(^{27}\) in our case yield function \(F(\sigma_{ij})\).

\(^{28}\) false position and secant method.

\(^{29}\) \(x\) as a quadratic function of \(y\).
In practice $b$ is the current best estimate of the root and the term:

$$\frac{f(b)}{f(a)} \left( \frac{f(a)}{f(c)} - 1 \right) \left( \frac{f(b)}{f(c)} - 1 \right) \left( \frac{f(b)}{f(a)} - 1 \right)$$

is a correction. Quadratic methods\(^{30}\) work well only when the function behaves smoothly. However, they run serious risk of giving bad estimates of the next root or causing floating point overflows, if divided by a small number

$$\left( \frac{f(a)}{f(c)} - 1 \right) \left( \frac{f(b)}{f(c)} - 1 \right) \left( \frac{f(b)}{f(a)} - 1 \right) \approx 0$$

Brent’s method prevents against this problem by maintaining brackets on the root and checking where the interpolation would land before carrying out the division. When the correction of type (104.92) would not land within bounds, or when the bounds are not collapsing rapidly enough, the algorithm takes a bisection step. Thus, Brent’s method combines the sureness of bisection with the speed of a higher order method when appropriate.

### 104.2.5 Singularities in the Yield Surface

#### 104.2.5.1 Corner Problem

Some yield criteria are defined with more than one yield surface\(^{31}\). We will restrict our attention to a two–surface yield criterion\(^{32}\). Koiter has shown in Koiter (1960) and Koiter (1953) that in the case when two yield surfaces are active, the plastic strain rate from equation (104.3) can be derived as follows:

$$d\epsilon_p^{ij} = d\lambda_{cone} m_{ij}^{cone} (\sigma_{ij}, q_s) + d\lambda_{cap} m_{ij}^{cap} (\sigma_{ij}, q_s)$$

(104.92)

where $m_{ij}^{cone} (\sigma_{ij}, q_s)$ and $m_{ij}^{cap} (\sigma_{ij}, q_s)$ are normals to the potential functions at a corner, which belongs to the yield functions that are active, i.e. $F_{cone}$ and $F_{cap}$. We now observe that we have two non–negative plastic multipliers $d\lambda_{cone}$ and $d\lambda_{cap}$ instead of one. We must require that at the end of

---

\(^{30}\) Newton’s method for example.

\(^{31}\) for example MRS-Lade yield criterion has two surfaces.

\(^{32}\) having in mind MRS-Lade cone-cap yield criterion.
\[ \sigma_{ij}^{n+1} \approx \sigma_{ij}^* \]

\[ F_{\text{cone}} \left( \sigma_{ij}^{n+1}, q_*^{n+1} \right) = 0 \] (104.93)

\[ F_{\text{cap}} \left( \sigma_{ij}^{n+1}, q_*^{n+1} \right) = 0 \] (104.94)

Figure 104.3: Pictorial representation of the corner point problem in computational elasto–plasticity: Yield surfaces with singular points

the loading step\(^{33}\), neither of the two yield functions is violated. These multipliers \(d\lambda_{\text{cone}} \) and \(d\lambda_{\text{cap}} \) can be determined from the conditions:

\(^{33}\text{after stress correction, i.e. return to the yield surface(s).} \)
noting that by virtue of equation (104.92) we have at the corner singular point:

\[ n^{+1} \sigma_{ij} = \text{pred}_{ij} \sigma_{ij} - d\lambda_{cone} E_{ijkl} \text{cone} m_{kl} - d\lambda_{cap} E_{ijkl} \text{cap} m_{kl} \]

(104.95)

### 104.2.5.2 Apex Problem

The apex problem, as depicted in Figure (104.4) is solved in an empirical fashion. Rather than facing the complexity of solving a complex differential geometry problem\(^{34}\) the stress point that is situated in the gray apex region is immediately returned to the apex point. In the case when the hardening rule for the cone portion has developed to the stage that it affects the size of that cone portion of the yield criterion and not the position of intersection with the hydrostatic axis, then all stress returns from any part of apex gray region will be to the apex point itself. This

\[ \sum_k d\lambda_k (\partial F_k / \partial \sigma_{ij}) \]

can be transformed into the integral equation $\text{d}e_{ij} = \int d\lambda \left( \partial F / \partial \sigma_{ij} \right)_{\text{around apex}}$ where the integration should be carried out infinitesimally close to, but in the vicinity of the apex point.

---

\(^{34}\)using Koiter’s work described in Koiter (1960) and Koiter (1953)
strategy was used by Crisfield (1987). Nevertheless, the problem of integrating the rate equations in
the apex gray region is readily solvable for the piecewise flat yield criteria\(^{\text{35}}\) by using Koiter’s conditions
as found in Koiter (1960) and Koiter (1953). The apex problem for yield criteria that are smooth and
differentiable everywhere except at the apex point, is solvable by means of differential geometry. Further
work is needed for solving the problem, when the yield surface is not piecewise flat in the apex vicinity.

104.2.5.3 Influence Regions in Meridian Plane

![Figure 104.5: Influence regions in the meridian plane for the cone/cap surface of the MRS-Lade material
model.](image)

In order to define which surface is active and which is not for the current state of stress, a simple
two dimensional analysis will be conducted. The fortunate fact for the MRS-Lade material model is
that such an analysis can be conducted in the \(p - q\) meridian plane, only, i.e. the value for \(\theta\) can be
"frozen". The concept is to calculate the stress invariants \(p, q\) and \(\theta\) for the current state of stress\(^{\text{36}}\),
calculate the position of the apex and corner points in \(p - q\) space for given the \(\theta\), calculate the two

\(^{\text{35}}\)Mohr - Coulomb for example.

\(^{\text{36}}\)by using equations (104.138) and (104.139) as defined in section (104.4.4).
dimensional gradients at these points, perform linear transformation of the current stress state\textsuperscript{37} to the new coordinate systems, and then check for the values of $p'_i$, $i = 1, 2, 3, 4$, where $p'_i$ is the transformed $p_i$ axis.

The angle $\psi$ is defined as the angle between the $p$ axis and the tangent to the potential function.

The gradients to the cone portion of the potential surface are defined as:

$$\frac{\partial Q_{\text{cone}}}{\partial p} = -n \eta_{\text{cone}}$$

$$\frac{\partial Q_{\text{cone}}}{\partial q} = g(\theta) \left( 1 + \frac{q}{q_a} \right)^m + \frac{g(\theta) m q (1 + \frac{q}{q_a})^{-1+m}}{q_a}$$

The gradients of the cap portion of the yield/potential surface is defined as:

$$\frac{\partial Q_{\text{cap}}}{\partial p} = \frac{2 (p - p_m)}{p_r^2}$$

$$\frac{\partial Q_{\text{cap}}}{\partial q} = \frac{2 g(\theta)^2 q \left( 1 + \frac{q}{q_a} \right)^{2m}}{f_r^2} + \frac{2 g(\theta)^2 m q^2 (1 + \frac{q}{q_a})^{-1+2m}}{f_r^2 q_a}$$

The vector of gradients in $p - q$ space is defined as:

$$\begin{bmatrix} \frac{\partial Q}{\partial p} \\ \frac{\partial Q}{\partial q} \end{bmatrix}$$

and the angle $\phi$ is calculated as:

$$\phi = \arctan \left( \frac{\frac{\partial Q}{\partial p}}{\frac{\partial Q}{\partial q}} \right) - 90^\circ$$

\textsuperscript{37}now in $p$, $q$ and $\theta$ space.
Care must be exercised with regard to which potential function is to be used in angle calculations. It should be mentioned that for the cap portion, the angle at the corner is $\phi = 0^\circ$, while at the tip of the cap, the angle is $\phi = -90^\circ$. If a new definition, as found in Ferrer (1992), is used for the cone potential function, where $n$ is variable and $n \rightarrow 0$ as $p \rightarrow \alpha p_{cap}$, then the corner gray region is empty.

The linear transformation\textsuperscript{38} between coordinate systems $p' - q'$ and $p - q$ is defined as:

\[
\begin{bmatrix}
p' \\
q'
\end{bmatrix}
= \begin{bmatrix}
\cos \phi & \sin \phi \\
-\sin \phi & \cos \phi
\end{bmatrix}
\begin{bmatrix}
p - tran_p \\
q - tran_q
\end{bmatrix}
\]

(104.98)

and by using that linear transformation, one can check the region where our current stress state, in $p$, $q$ and $\theta$ space, belongs. Figure (104.5) depicts the transformation scheme and the new coordinate systems at three important points\textsuperscript{39}.

### 104.3 A Forward Euler (Explicit) Algorithm

The explicit algorithm (Forward Euler) is based on using the starting point (the state stress $\sigma_{ij}^n$ and internal variable space $q^*_n$ on the yield surface) for finding all the relevant derivatives and variables.

The Explicit algorithm can be derived by starting from a first order Taylor expansion about starting point $(\sigma_{ij}^0, q^*_n)$:

\[
F_{\text{new}} = F_{\text{old}} + \left. \frac{\partial F}{\partial \sigma_{mn}} \right|_n d(\eta_{\sigma_{mn}}) + \left. \frac{\partial F}{\partial q^*_s} \right|_n dq^*_s = 0
\]

(104.99)

From the differential form of equation (104.16) it follows:

\[
d \left( f \sigma_{mn} \right) = E_{mnq} d(\epsilon_{pq}) - E_{mpq} d(\epsilon_{pq}^p) = 0
\]

so that equation (104.99) becomes:

\[
n_{\eta_{\sigma_{mn}} E_{mnq}} d\epsilon_{pq} - n_{\eta_{\sigma_{mn}} E_{mpq}} d\lambda m_{pq} + \xi_\ast h_\ast d\lambda = 0
\]

\textsuperscript{38} translation and rotation.

\textsuperscript{39} at the apex point, corner point and the cap tip point.
and it follows, after solving for $d\lambda$

$$d\lambda = \frac{n_{mn}}{\text{cross} m_{ab} E_{abcd} \text{cross} n_{cd} - \xi h_A} E_{mnpq} \, d\epsilon_{pq}$$

With this solution for $d\lambda$ one can obtain the increments in stress tensor and internal variables as

$$d\sigma_{mn} = E_{mnpq} \, d\epsilon_{pq} - E_{mnpq} \, \frac{n_{rs} E_{rstu} \, d\epsilon_{tu}}{n_{ab} E_{abcd} \, n_{cd} - \xi A h_A} n_{mpq}$$

(104.100)

$$dq_A = \left( \frac{n_{mn}}{\text{cross} m_{ab} E_{abcd} \text{cross} n_{cd} - \xi B h_B} \right) h_A$$

(104.101)

where $n()$ denotes the starting elastic–plastic point for that increment. It should be noted that the explicit algorithm performs only one step of the computation and does not check on the equilibrium of the obtained solutions. This usually results in the slow drift of the stress–internal variable point from the yield surface for monotonic loading. It also results in spurious plastic deformations during elastic unloading for cyclic loading–unloading.

### 104.3.1 Continuum Tangent Stiffness Tensor.

The continuum tangent stiffness tensor ($^{\text{cont}}E_{pqmn}$) is obtained from the explicit (forward Euler) integration procedure (Jeremić and Sture, 1997):

$$^{\text{cont}}E_{pqmn} = E_{pqmn} - \frac{E_{pqkl} n_{mk} n_{lij} E_{ijkl}}{n_{ot} E_{ots} n_{mrs} - \xi A h_A}$$

(104.102)

It is important to note that continuum tangent stiffness ($^{\text{cont}}E_{pqmn}$) possesses minor symmetries ($^{\text{cont}}E_{pqmn} = ^{\text{cont}}E_{pqn} = ^{\text{cont}}E_{pqmn}$), while major symmetry ($^{\text{cont}}E_{pqmn} = ^{\text{cont}}E_{mpnq}$), is only retained for associated elastic–plastic materials, when $n_{ij} \equiv m_{ij}$.

### 104.4 A Backward Euler (Implicit) Algorithm

In previous sections, the general theory of elasto–plasticity was presented. The accuracy and stability for the general Midpoint rule algorithm has been shown. In this chapter, the focus is on the Backward Euler algorithm, which is derived from the general Midpoint algorithm by setting $\alpha = 1$. The advantage of the Backward Euler scheme over other midpoint schemes is that the solution is sought by using the normal\(^{40} m_{ij} = \frac{\partial Q}{\partial \sigma_{ij}}\) at the final stress state. By implicitly assuming that such a stress state exists, the Backward
Euler scheme is guaranteed to provide a solution, despite the size of the strain step\textsuperscript{41}. However, it was shown in section (104.2.3.1) that the Backward Euler algorithm is only accurate to the first order.

The full implicit Backward Euler algorithm is based on the equation:

\[ n^{+1}_{ij} = \text{pred}_{ij} - \Delta \lambda \ E_{ijkl} \ n^{+1}_{mkl} \]  

(104.103)

where \( \text{pred}_{ij} = E_{ijkl} \ \epsilon_{kl} \) is the elastic trial stress state, \( Q \) is the plastic potential function and \( \nabla_{\sigma_{kl}} \bigg|_{n+1} \) is the gradient to the plastic potential function in the stress space at the final stress position, and

\[ \text{pred}_{ij} = n_{ij} + E_{ijkl} \ \text{pred}_{\Delta \epsilon_{kl}} \]  

(104.104)

is the elastic predicted (trial) stress state.

An initial estimate for the stress \( \nabla^{+1}_{ij} \) can be obtained using various other methods. This estimate generally does not satisfy the yield condition, so some kind of iterative scheme is necessary to return the stress to the yield surface.

### 104.4.1 Single Vector Return Algorithm.

If the predictor stress \( \text{pred}_{ij} \) is not in a corner or apex gray regions, a single vector return to the yield surface is possible. In order to derive such a scheme for a single vector return algorithm, a tensor of residuals \( r_{ij} \) will be defined as \textsuperscript{42}:

\[ r_{ij} = \sigma_{ij} - \left( \text{pred}_{ij} - \Delta \lambda \ E_{ijkl} \ m_{kl} \right) \]  

(104.105)

This tensor represents the difference between the current stress state \( \sigma_{ij} \) and the Backward Euler stress state \( \text{pred}_{ij} - \Delta \lambda \ E_{ijkl} \ m_{kl} \).

The trial stress state \( \text{pred}_{ij} \) is kept fixed during the iteration process. The first order Taylor series expansion can be applied to Equation 104.105 to obtain the new residual \( \text{new}_{r_{ij}} \) from the old one \( \text{old}_{r_{ij}} \)

\[ \text{new}_{r_{ij}} = \text{old}_{r_{ij}} + d \sigma_{ij} + d(\Delta \lambda) \ E_{ijkl} \ m_{kl} + \Delta \lambda \ E_{ijkl} \left( \frac{\partial m_{kl}}{\partial \sigma_{mn}} d \sigma_{mn} + \frac{\partial m_{kl}}{\partial q_A} d q_A \right) \]  

(104.106)

where \( d \sigma_{ij} \) is the change in \( \sigma_{ij} \), \( d(\Delta \lambda) \) is the change in \( \Delta \lambda \), and \( \frac{\partial m_{kl}}{\partial \sigma_{mn}} d \sigma_{mn} + \frac{\partial m_{kl}}{\partial q_A} d q_A \) is the change in \( m_{kl} \) at \( q_A \). The goal is let \( \text{new}_{r_{ij}} = \emptyset \), so one can write

\[ \emptyset = \text{old}_{r_{ij}} + d \sigma_{ij} + d(\Delta \lambda) \ E_{ijkl} \ m_{kl} + \Delta \lambda \ E_{ijkl} \left( \frac{\partial m_{kl}}{\partial \sigma_{mn}} d \sigma_{mn} + \frac{\partial m_{kl}}{\partial q_A} d q_A \right) \]  

(104.107)

\textsuperscript{41}large strain step increments were tested, the scheme converged to the solution even for deviatoric strain steps of 20% in magnitude.

\textsuperscript{42}By default at increment \( n + 1 \), and \( n^{+1}(\cdot) \) is omitted for simplicity.
Similarly,
\[ q_A = n q_A + \Delta \lambda h_A \] (104.108)

\( r_A \) will be defined as:
\[ r_A = q_A - (n q_A + \Delta \lambda h_A) \] (104.109)

and \( n q_A \) is kept fixed during iteration, that
\[ \emptyset = \text{old} r_A + d q_A - d(\Delta \lambda) h_A - \Delta \lambda \left( \frac{\partial h_A}{\partial \sigma_{ij}} d \sigma_{ij} + \frac{\partial h_A}{\partial q_B} d q_B \right) \] (104.110)

From equation 104.107 and 104.110, one obtains
\[
\begin{bmatrix}
\mathcal{I} + \Delta \lambda \mathbb{E} \frac{\partial m_{kl}}{\partial \sigma_{mn}} & \Delta \lambda \mathbb{E} \frac{\partial m_{kl}}{\partial q_B} \\
-\Delta \lambda \frac{\partial h_A}{\partial \sigma_{ij}} & \delta_{AB} - \Delta \lambda \frac{\partial h_A}{\partial q_B}
\end{bmatrix}
\begin{bmatrix}
d \sigma_{mn} \\
d q_B
\end{bmatrix}
+ d(\Delta \lambda)
\begin{bmatrix}
E_{ijkl} m_{kl} \\
-h_A
\end{bmatrix}
= \emptyset
\] (104.111)

Since \( f(\sigma_{ij}, q_A) = 0 \), one obtains
\[ \emptyset = \text{old} f + n mn d \sigma_{mn} + \xi_B d q_B \] (104.112)

From equations 104.111 and 104.112,
\[
\begin{bmatrix}
\mathcal{I} + \Delta \lambda \mathbb{E} \frac{\partial m_{kl}}{\partial \sigma_{mn}} & \Delta \lambda \mathbb{E} \frac{\partial m_{kl}}{\partial q_B} \\
-\Delta \lambda \frac{\partial h_A}{\partial \sigma_{ij}} & \delta_{AB} - \Delta \lambda \frac{\partial h_A}{\partial q_B}
\end{bmatrix}
\begin{bmatrix}
E_{ijkl} m_{kl} \\
-h_A
\end{bmatrix}
= \text{old} f
\] (104.113)

The iteration of \( \Delta \lambda \) is then
\[ \Delta \lambda^{k+1} = \Delta \lambda^k + d(\Delta \lambda)^k \] (104.114)

The iterative procedure is continued until the yield criterion \( f = 0, \| r_{ij} \| = \emptyset \), and \( \| r_A \| = \emptyset \) are satisfied within some tolerances at the final stress state \(^{43}\).

In Equation 104.113, the generalized matrix \( \mathbb{C} \), which is defined by
\[
\mathbb{C} = \left[ \begin{array}{cc}
\mathcal{I} + \Delta \lambda \mathbb{E} \frac{\partial m_{kl}}{\partial \sigma_{mn}} & \Delta \lambda \mathbb{E} \frac{\partial m_{kl}}{\partial q_B} \\
-\Delta \lambda \frac{\partial h_A}{\partial \sigma_{ij}} & \delta_{AB} - \Delta \lambda \frac{\partial h_A}{\partial q_B}
\end{array} \right]^{-1}
\] (104.115)

\(^{43}\| \| \) is some normal of the tensor
plays an important role in the implicit algorithm. It should be mentioned here that the above definition is a simplified expression for very general model with various isotropic and kinematic hardening. Specifically, if there is no hardening,

$$C = \left[ I_{ijmn}^s + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \sigma_{mn}} \right]^{-1}$$  \hspace{3cm} (104.116)

If there is only one isotropic internal variable \( q \),

$$C = \left[ I_{ijmn}^s + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \sigma_{mn}} \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial q} \right]^{-1}$$  \hspace{3cm} (104.117)

For only one kinematic internal variable \( \alpha_{ij} \),

$$C = \left[ I_{ijmn}^s + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \sigma_{mn}} \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \alpha_{ij}} \right]^{-1}$$  \hspace{3cm} (104.118)

For one isotropic variable \( q \) and one kinematic variable \( \alpha_{ij} \),

$$C = \left[ I_{ijmn}^s + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \sigma_{mn}} \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \alpha_{ij}} \right]^{-1}$$  \hspace{3cm} (104.119)

or for two kinematic variables \( z_{ij} \) and \( \alpha_{ij} \),

$$C = \left[ I_{ijmn}^s + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \sigma_{mn}} \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial z_{mn}} \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \alpha_{ij}} \right]^{-1}$$  \hspace{3cm} (104.120)

If we define

$$n = \{ n_{mn} \} $$ \hspace{3cm} (104.121)

$$m = \left\{ \begin{array}{c} E_{ijkl} m_{kl} \\ -h_A \end{array} \right\} $$ \hspace{3cm} (104.122)

$$old \mathbf{r} = \left\{ \begin{array}{c} old \mathbf{r}_{ij} \\ old \mathbf{r}_A \end{array} \right\} $$ \hspace{3cm} (104.123)

Equation 104.114 can be simplified as

$$d(\Delta \lambda) = \frac{old \mathbf{f} - n^T C old \mathbf{r}}{n^T C M}$$  \hspace{3cm} (104.124)

and

$$\left\{ \begin{array}{l} d\sigma_{mn} \\ dq_B \end{array} \right\} = -C \left( old \mathbf{r} + d(\Delta \lambda) m \right)$$  \hspace{3cm} (104.125)
104.4.2 Backward Euler Algorithms: Starting Points

Some remarks are necessary in order to clarify the Backward Euler Algorithm. It is a well known fact that the rate of convergence of the Newton-Raphson Method, or even obtaining convergence at all, is closely tied to the starting point for the iterative procedure. Bad initial or starting points might lead our algorithm to an oscillating solution, i.e. the algorithm does not converge. In the following, starting points for the Newton-Raphson iterative procedure will be established for one- and two-vector return algorithms.

104.4.2.1 Single Vector Return Algorithm Starting Point.

One of the proposed starting points (Crisfield, 1991) uses the normal at the elastic trial point \( \text{pred}_\sigma_{ij} \).

A first order Taylor expansion about point \( \text{pred}_\sigma_{ij} \) yields:

\[
\text{pred}_F^{\text{new}} = \text{pred}_F^{\text{old}} + \frac{\partial F}{\partial \sigma_{mn}} \bigg|_{\text{pred}} \left( \text{pred}_\sigma_{mn} \right) + \frac{\partial F}{\partial q_A} \bigg|_{\text{pred}} dq_A = \\
= \text{pred}_F^{\text{old}} + \text{pred}_n_{mn} d\sigma_{mn} + \xi_A h_A d\lambda = 0
\]

(104.126)

It is assumed that the total incremental strain \( \epsilon_{kl} \) is applied in order to reach the point \( \text{pred}_\sigma_{ij} \), i.e. \( \text{pred}_\sigma_{ij} = E_{ijkl} \epsilon_{kl} \) so that any further stress "relaxation" toward the yield surface takes place under zero total strain condition \( \epsilon_{kl} = 0 \). From the differential form of equation (104.16) it follows:

\[
d\left( \text{pred}_\sigma_{mn} \right) = E_{mnpq} \left( d\left( \text{pred}_\epsilon_{pq} \right) - d\left( \text{pred}_\epsilon_{pq} \right) \right) = \\
= -E_{mnpq} d\left( \text{pred}_\epsilon_{pq} \right) = -E_{mnpq} d\lambda \left( \text{pred}_\sigma_{pq} \right)
\]

and equation (104.126) becomes:

\[
\text{pred}_F^{\text{old}} - \text{pred}_n_{mn} E_{mnpq} d\lambda \text{pred}_m_{pq} + \xi_A h_A d\lambda = 0
\]

and it follows:

\[
d\lambda = \frac{\text{pred}_F^{\text{old}}}{\text{pred}_n_{mn} E_{mnpq} \text{pred}_m_{pq} - \xi_A h_A}
\]

With this solution for \( d\lambda \) we can obtain the starting point for the Newton-Raphson iterative procedure:

\[
\text{start}_\sigma_{mn} = E_{mnpq} \text{pred}_\epsilon_{pq} - E_{mnpq} \frac{\text{pred}_F^{\text{old}}}{\text{pred}_n_{mn} E_{mnpq} \text{pred}_m_{pq} - \xi_A h_A} \text{pred}_m_{pq}
\]

(104.127)

\( ^{44} \)I have named this scheme as semi Backward Euler scheme.
This starting point in six dimensional stress space will in general not satisfy the yield condition \( F = 0 \), but it will provide a good initial guess for the upcoming Newton-Raphson iterative procedure.

It should be mentioned, however, that this scheme for returning to the yield surface is the well known Radial Return Algorithm, if the yield criterion under consideration is of the von Mises type. In the special case the normal at the elastic trial point \( \sigma_{ij}^{pred} \) coincides with the normal at the final stress state \( \sigma_{ij}^{n+1} \), the return is exact, i.e. the yield condition is satisfied in one step.

Another possible and readily available starting point can be obtained by applying one Forward Euler step\(^{45} \). To be able to use the Forward Euler integration scheme, an intersection point has to be found. The procedure for calculating intersection points is given in section (104.2.4).

A first order Taylor expansion about intersection point \( \sigma_{ij}^{cross} \) yields:

\[
F_{\text{new}} = F_{\text{old}} + \left. \frac{\partial F}{\partial \sigma_{mn}} \right|_{\text{cross}} d \sigma_{mn} + \left. \frac{\partial F}{\partial q_A} \right|_{\text{cross}} dq_A = 0
\]

(104.128)

From the differential form of equation (104.16) it follows:

\[
d \left( f E \sigma_{mn} \right) = E_{mn pq} \left( d \left( \epsilon_{pq} \right) - d \left( \epsilon_{pq}^p \right) \right) =
\]

\[
= E_{mn pq} d \left( \epsilon_{pq} \right) - E_{mn pq} d \left( \epsilon_{pq}^p \right) = E_{mn pq} d \left( \epsilon_{pq} \right) - E_{mn pq} d \lambda \left( \sigma_{pq}^{cross} \right)
\]

and equation (104.128) becomes:

\[
- \sigma_{mn}^{cross} E_{mn pq} d \epsilon_{pq} - \sigma_{mn}^{cross} E_{mn pq} d \lambda \sigma_{pq}^{cross} + \xi_A h_A d \lambda = 0
\]

and it follows

\[
d \lambda = \frac{\sigma_{mn}^{cross} E_{mn pq} d \epsilon_{pq}}{\sigma_{mn}^{cross} E_{mn pq} \sigma_{pq}^{cross} - \xi_A h_A}
\]

With this solution for \( d \lambda \) we can obtain the starting point for the Newton-Raphson iterative procedure

\[
\sigma_{mn}^{start} = E_{mn pq} d \epsilon_{pq} - E_{mn pq} \sigma_{mn}^{cross} E_{rstu} d \epsilon_{tu} \sigma_{pq}^{cross} - \xi_A h_A
\]

(104.129)

\(^{45}\)or more steps for really large strain increments, for example over 10% in deviatoric direction. What has actually been done is to divide the \( \theta \) region into several parts and depending on the curvature of the yield surface in deviatoric plane, use different schemes and different number of subincrements (the more curved, the more subincrements) to get the first, good initial guess. In the region around \( \theta = 0 \), one step of the semi Backward Euler scheme is appropriate, but close to \( \theta = \pi/3 \) the Forward Euler subincrementation works better.
This starting point in six–dimensional stress space will again not satisfy the yield condition $F = 0$, but will provide a good initial estimate for the upcoming Newton-Raphson iterative procedure.

### 104.4.3 Consistent Tangent Stiffness Tensor

The final goal in deriving the Backward Euler scheme for integration of elasto–plastic constitutive equations is to use that scheme in finite element computations. If the Newton – Raphson iterative scheme is used at the global equilibrium level then the use of the so called traditional tangent stiffness tensor $E^{ep}_{ijkl}$ destroys the quadratic rate of asymptotic convergence of the iterative scheme. In order to preserve such a quadratic rate, a consistent, also called algorithmic, tangent stiffness tensor is derived. The consistent tangent stiffness tensor make use of derivatives of direction normal to the potential function, and they are derived at the final, final at each iteration, that converges to the final stress point on the yield surface, stress point. The traditional forward scheme has a constant derivative, $m_{ij}$ that is evaluated at the intersection point.

It appears that Simo and Taylor (1985) and Runesson and Samuelsson (1985) have first derived the consistent tangent stiffness tensor. Other interesting articles on the subject can be found in Simo and Taylor (1986), Simo and Govindjee (1988), Jetteur (1986), Braudel et al. (1986), Crisfield (1987), Ramm and Matzenmiller (1988) and Mitchell and Owen (1988). As a consequence of consistency, the use of the consistent tangent stiffness tensor significantly improves the convergence characteristics of the overall equilibrium iterations, if a Newton - Raphson scheme is used for the latter. Use of the consistent tangent stiffness tensor yields a quadratic convergence rate of Newton - Raphson equilibrium iterations. In what follows, two derivations are given, namely the consistent tangent stiffness tensor for single– and two–vector return algorithms.

The concept of consistent linearization was introduced by Hughes and Pister (1978), while detailed explanation is given by Simo and Hughes (1998). The consistent tangent stiffness leads to quadratic convergence rates at global level.

It should be mentioned that there are various ‘equivalent’ forms of consistent tangent stiffness depending on the specific implicit algorithm equations. For instance, Simo and Hughes (1998), and Belytschko et al. (2001) derived the consistent tangent stiffness by taking current plastic strain as unknown and seeking its derivatives in the stress space; Pérez-Foguet and Huerta (1997) and Pérez-Foguet et al. (2000) used the numerical differentiation to calculate the consistent tangent stiffness in a

---

46 except for the yield criteria that have flat yield surfaces ( in the stress invariant space) so that the first order Taylor linear expansion, is exact.

47 the one obtained with the Forward Euler method, i.e. where parameter $\alpha = 0$.

48 $m_{ij} = \partial Q/\partial \sigma_{ij}$, i.e. $\partial m_{ij}/\partial \sigma_{kl} = \partial^2 Q/\partial \sigma_{ij} \partial \sigma_{kl}$.
compact matrix-vector form; Choi (2004) adopted the compact matrix-vector form by Pérez-Foguet and Huerta (1997) and Pérez-Foguet et al. (2000) but taking current plastic strain as unknown and seeking its derivatives in the elastic strain space. Slightly different from the above strategies, in this work the implicit algorithm is adopting the traditional form but taking current stress as unknown and seeking its derivatives in the stress space. Provided these differences, the consistent tangent stiffness in this work is slightly different from those in the above work.

104.4.3.1 Single Vector Return Algorithm.

In implicit algorithm, a very important advantage is that it may lead to consistent (algorithmic) tangent stiffness (Equation 104.137). The concept of consistent linearization was introduced in Hughes and Pister (1978), more details on consistent tangent stiffness were explained in Simo and Hughes (1998). The consistent tangent stiffness leads to quadratic convergence rates at global level.

It should be mentioned that there are various ‘equivalent’ forms of consistent tangent stiffness depending on the specific implicit algorithm equations. For instance, Simo and Hughes (1998), and Belytschko et al. (2001) derived the consistent tangent stiffness by taking current plastic strain as unknown and seeking its derivatives in the stress space; Pérez-Foguet and Huerta (1997) and Pérez-Foguet et al. (2000) used the numerical differentiation to calculate the consistent tangent stiffness in a compact matrix-vector form; Choi (2004) adopted the compact matrix-vector form by Pérez-Foguet and Huerta (1997) and Pérez-Foguet et al. (2000) but taking current plastic strain as unknown and seeking its derivatives in the elastic strain space. Slightly different from the above strategies, in this work (section 104.4) the implicit algorithm is adopting the traditional form but taking current stress as unknown and seeking its derivatives in the stress space. Provided these differences, the consistent tangent stiffness in this work is slightly different from those in the above work. The detail derivation will be followed.

When seeking the algorithmic tangent stiffness, we look into the explicit expression of dσ_{ij}/d\epsilon_{mn}^{pred}.

At the same time, the internal variables are initialized the values at the previous time step, in other words, they are fixed within the time step when seeking the algorithmic tangent stiffness.

Linearize Equation 104.103, one obtains

\[ d\sigma_{ij} = E_{ijkl} \, d\epsilon_{kl}^{pred} - d(\Delta \lambda) \, E_{ijkl} \, m_{kl} - \Delta \lambda \, E_{ijkl} \left( \frac{\partial m_{kl}}{\partial \sigma_{mn}} \, d\sigma_{mn} + \frac{\partial m_{kl}}{\partial q_A} \, dq_A \right) \]  \hspace{1cm} (104.130)

Similarly, linearize Equation 104.108, one obtains

\[ dq_A = d(\Delta \lambda) \, h_A + \Delta \lambda \left( \frac{\partial h_A}{\partial \sigma_{ij}} \, d\sigma_{ij} + \frac{\partial h_A}{\partial q_B} \, dq_B \right) \]  \hspace{1cm} (104.131)
From equation 104.130 and 104.131, one obtains
\[
\begin{bmatrix}
I_{ijmn} + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \sigma_{mn}} + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial q_A} \\
-\Delta \lambda \frac{\partial h_A}{\partial \sigma_{ij}} - \Delta \lambda \frac{\partial h_A}{\partial q_B} \\
\end{bmatrix}
\begin{bmatrix}
d\sigma_{mn} \\
dq_B \\
\end{bmatrix} = d(\Delta \lambda) \begin{bmatrix}
E_{ijkl} m_{kl} \\
h_A \\
\end{bmatrix}
\begin{bmatrix}
E_{ijkl} d\epsilon_{kl}^{\text{pred}} \\
0 \\
\end{bmatrix}
\] (104.132)

If one uses the definitions of 104.115, 104.122 and 104.121, Equation 104.132 can be simplified to
\[
C^{-1} \begin{bmatrix}
d\sigma_{mn} \\
dq_B \\
\end{bmatrix} + d(\Delta \lambda) m = \begin{bmatrix}
E_{ijkl} d\epsilon_{kl}^{\text{pred}} \\
0 \\
\end{bmatrix}
\] (104.133)

Linearize the yield function \( f(\sigma_{ij}, q_A) = 0 \), one obtains
\[
n_{mn} d\sigma_{mn} + \xi_B dq_B = 0
\] (104.134)

or in a simplified form
\[
n^T \begin{bmatrix}
d\sigma_{mn} \\
dq_B \\
\end{bmatrix} = 0
\] (104.135)

From Equations 104.133 and 104.135, one obtain
\[
d(\Delta \lambda) = \frac{n^T C}{n^T C m} \begin{bmatrix}
E_{ijkl} d\epsilon_{kl}^{\text{pred}} \\
0 \\
\end{bmatrix}
\] (104.136)

Substitute expression 104.136 into 104.133, one obtains
\[
\begin{bmatrix}
d\sigma_{ij} \\
dq_A \\
\end{bmatrix} = \left( C - \frac{C m n^T C}{n^T C m} \right) \begin{bmatrix}
E_{ijkl} d\epsilon_{kl}^{\text{pred}} \\
0 \\
\end{bmatrix}
\] (104.137)

This equation gives the explicit expression of the consistent tangent stiffness \( d\sigma_{ij}/d\epsilon_{mn}^{\text{pred}} \) for the implicit algorithm.

From section 104.4, if there are interactions between internal variables, the implicit algorithm will become very complicated. Simple models (e.g. von Mises model, or sometimes termed as \( J_2 \) model) have been proved efficient and good performance by the implicit algorithm (Simo and Hughes, 1998). Evidently, the implicit algorithm is mathematically based on the Newton-Raphson nonlinear equation solving method as well as the Eulerian backward integration method. Theoretically, the Newton-Raphson method may have quadratic convergence rate. However, Newton-Raphson method is not unconditional stable, and sometimes the iteration will diverge (Press et al., 1988a). Any bad starting point, non-continuous derivatives around solution, high nonlinearity, and interactions between internal variables, will
deteriorate the implicit algorithm performance. A complicated model cannot guarantee good performance or quadratic convergence by the implicit algorithm Crisfield (1997a). The task to obtain the analytical expressions (Equations 104.116 to 104.120) may prove exceeding laborious for complicated plasticity models Simo and Hughes (1998).

### 104.4.4 Gradients to the Potential Function

In the derivation of the Backward Euler algorithm and the Consistent Tangent Matrix it is necessary to derive the first and the second derivatives of the potential function. The function $Q$ is the function of the stress tensor $\sigma_{ij}$ and the plastic variable tensor $q_A$. Derivatives with respect to the stress tensor $\sigma_{ij}$ and plastic variable tensor $q_A$ are given here. It is assumed that any stress state can be represented with the three stress invariants $p, q$ and $\theta$ given in the following form:

$$
\begin{align*}
  p &= -\frac{1}{3} I_1 \\
  q &= \sqrt{3 J_{2D}} \\
  \cos 3\theta &= \frac{3\sqrt{3}}{2} \frac{J_{3D}}{\sqrt{(J_{2D})^3}} \\
  I_1 &= \sigma_{kk} \\
  J_{2D} &= \frac{1}{2} s_{ij}s_{ij} \\
  J_{3D} &= \frac{1}{3} s_{ij}s_{jk}s_{ki} \\
  s_{ij} &= \sigma_{ij} - \frac{1}{3} \sigma_{kk}\delta_{ij}
\end{align*}
$$

(104.138)

Stresses are here chosen as positive in tension. The definition of Lode's angle $\theta$ in equation (104.138) implies that $\theta = 0$ defines the meridian of conventional triaxial extension (CTE), while $\theta = \pi/3$ denotes the meridian of conventional triaxial compression (CTC).

The Potential Function is given in the following form:

$$
Q = Q(p, q, \theta)
$$

(104.140)

The complete derivation of the closed form gradients is given in Appendix 703.

#### 104.4.4.1 Analytical Gradients

The first derivative of the function $Q$ in stress space is:

$$
\frac{\partial Q}{\partial \sigma_{ij}} = \frac{\partial Q}{\partial p} \frac{\partial p}{\partial \sigma_{ij}} + \frac{\partial Q}{\partial q} \frac{\partial q}{\partial \sigma_{ij}} + \frac{\partial Q}{\partial \theta} \frac{\partial \theta}{\partial \sigma_{ij}}
$$

(104.141)
and subsequently the first derivatives of the chosen stress invariants are

\[
\frac{\partial p}{\partial \sigma_{ij}} = -\frac{1}{3} \delta_{ij} \tag{104.142}
\]

\[
\frac{\partial q}{\partial \sigma_{ij}} = \frac{3}{2} \frac{1}{q^2} s_{ij} \tag{104.143}
\]

\[
\frac{\partial \theta}{\partial \sigma_{ij}} = \frac{3}{2} \cos(3\theta) \frac{1}{q^2 \sin(3\theta)} s_{ij} - \frac{9}{2} \frac{1}{q^3 \sin(3\theta)} t_{ij} \tag{104.144}
\]

where:

\[
t_{ij} = \frac{\partial J_{3D}}{\partial \sigma_{ij}}
\]

The second derivative of the function \( Q \) in stress space is

\[
\frac{\partial^2 Q}{\partial \sigma_{pq} \partial \sigma_{mn}} = \left( \frac{\partial^2 Q}{\partial p^2} \frac{\partial p}{\partial \sigma_{mn}} + \frac{\partial^2 Q}{\partial q^2} \frac{\partial q}{\partial \sigma_{mn}} + \frac{\partial^2 Q}{\partial \theta^2} \frac{\partial \theta}{\partial \sigma_{mn}} \right) \frac{\partial p}{\partial \sigma_{pq}} + \frac{\partial Q}{\partial \sigma_{pq}} \frac{\partial^2 p}{\partial \sigma_{mn}} +
\]

\[
+ \left( \frac{\partial^2 Q}{\partial q \partial p} \frac{\partial p}{\partial \sigma_{mn}} + \frac{\partial^2 Q}{\partial q \partial \theta} \frac{\partial \theta}{\partial \sigma_{mn}} \right) \frac{\partial q}{\partial \sigma_{pq}} + \frac{\partial Q}{\partial \sigma_{pq}} \frac{\partial^2 q}{\partial \sigma_{mn}} +
\]

\[
+ \left( \frac{\partial^2 Q}{\partial \theta \partial p} \frac{\partial p}{\partial \sigma_{mn}} + \frac{\partial^2 Q}{\partial \theta \partial \theta} \frac{\partial \theta}{\partial \sigma_{mn}} \right) \frac{\partial \theta}{\partial \sigma_{pq}} + \frac{\partial Q}{\partial \sigma_{pq}} \frac{\partial^2 \theta}{\partial \sigma_{mn}} \tag{104.145}
\]

and the second derivatives of the stress invariants are

\[
\frac{\partial^2 p}{\partial \sigma_{pq} \partial \sigma_{mn}} = 0 \tag{104.146}
\]
\[
\frac{\partial^2 q}{\partial \sigma_{pq} \partial \sigma_{mn}} = \frac{3}{2} q \left( \delta_{pm} \delta_{nq} - \frac{1}{3} \delta_{pq} \delta_{nm} \right) - \frac{9}{4} q^3 s_{mn} s_{pq}
\]  
(104.147)

\[
\frac{\partial^2 \theta}{\partial \sigma_{pq} \partial \sigma_{mn}} =
\]

\[
- \left( \frac{9}{2} \frac{\cos 3\theta}{q^4 \sin (3\theta)} + \frac{27}{4} \frac{\cos 3\theta}{q^4 \sin^3 3\theta} \right) s_{pq} s_{mn} + \frac{81}{4} q^2 \frac{1}{q^5 \sin^3 3\theta} s_{pq} t_{mn} + \\
+ \left( \frac{81}{4} \frac{1}{q^5 \sin 3\theta} + \frac{81}{4} \frac{\cos^2 3\theta}{q^5 \sin 3\theta} \right) t_{pq} s_{mn} - \frac{243}{4} \frac{\cos 3\theta}{q^6 \sin^3 3\theta} t_{pq} t_{mn} + \\
+ \frac{3}{2} \frac{\cos (3\theta)}{q^2 \sin (3\theta)} p_{pqmn} + \frac{9}{2} \frac{1}{q^3 \sin (3\theta)} w_{pqmn}
\]  
(104.148)

where:

\[
w_{pqmn} = \frac{\partial v_{pq}}{\partial \sigma_{mn}} = s_{np} \delta_{qm} + s_{qm} \delta_{np} - \frac{2}{3} s_{qp} \delta_{nm} - \frac{2}{3} \delta_{pq} s_{mn}
\]

\[
p_{pqmn} = \frac{\partial s_{pq}}{\partial \sigma_{mn}} = \left( \delta_{mp} \delta_{nq} - \frac{1}{3} \delta_{pq} \delta_{mn} \right)
\]

and:

Another important gradient is:

\[
\frac{\partial^2 Q}{\partial \sigma_{ij} \partial q_A} = \frac{\partial m_{ij}}{\partial q_A} =
\]

\[
= \frac{\partial^2 Q}{\partial q_A \partial \sigma_{ij}} + \frac{\partial Q}{\partial q_A} \frac{\partial \sigma_{ij}}{\partial q_A} + \frac{\partial Q}{\partial \sigma_{ij}} \frac{\partial q_A}{\partial \sigma_{ij}}
\]

\[
= \frac{\partial^2 Q}{\partial p \partial q_A} \frac{\partial \sigma_{ij}}{\partial q_A} + \frac{\partial^2 Q}{\partial q \partial q_A} \frac{\partial \sigma_{ij}}{\partial q_A} + \frac{\partial^2 Q}{\partial \theta \partial q_A} \frac{\partial \sigma_{ij}}{\partial q_A} + \frac{\partial \sigma_{ij}}{\partial q_A} \frac{\partial q_A}{\partial \sigma_{ij}}
\]  
(104.149)
104.4.4.2 Finite Difference Gradients

After having developed the closed form, analytical derivatives\(^{49}\) the author of this thesis asked himself: "is there a simpler way of finding these derivatives?" One of the proposed ways to check the analytical solution is found in Dennis and Schnabel (1983). Dennis and Schnabel propose the finite difference method for approximating derivatives if these derivatives are not analytically available and as a tool to check your analytical derivatives if they are derived.

Another good reason for developing alternative gradients is that for \(\theta = 0, \pi/3\) gradients are not defined, i.e. indefinite terms as \(0/0\) are appearing. One possible solution is the use of l'Hospital's rule. This has been done in Perić (1991). The solution to the problem in this work went in a different direction, i.e. instead of aiming for the analytical form, numerical derivatives are derived.

We should recall that for a function \(f\) of a single variable, the finite difference approximation to \(f'(x)\), by using forward finite difference approach, is given by:

\[
a = \frac{f(x + h) - f(x)}{h}
\]  

(104.150)

where \(h\) is a vanishingly small quantity. The same definition was used in deriving the finite difference approximation for the first derivative of the yield function \(F\) and potential function \(Q\). The first derivative of \(F\) (or \(Q\)) with respect to the stress tensor \(\sigma_{ij}\) for diagonal elements is\(^{50}\):

\[
\text{approx.} F_{,ii} = \frac{F(\sigma_{ii} + h_{ii}) - F(\sigma_{ii})}{h_{ii}}
\]  

(104.151)

and for non-diagonal elements\(^{51}\):

\[
\text{approx.} F_{,ij} = \frac{F(\sigma_{ij} + h_{ij} + h_{ji}) - F(\sigma_{ij})}{2h_{ij}}
\]  

(104.152)

where \(h_{ij}\) is the step size which, because of finite precision arithmetic, is a variable\(^{52}\).

The accuracy of the finite difference approximation to the analytical derivatives is closely bound to the step size \(h_{ij}\). It is suggested in Dennis and Schnabel (1983)[section 5.4.] that for functions given by the simple formula, the number \(h\) should be \(h = \sqrt{\text{macheps}}\), while for more complicated

---

\(^{49}\)see Appendix (703).

\(^{50}\)no sum convention implied, just the position of the element.

\(^{51}\)since the stress tensor \(\sigma_{ij}\) is symmetric, change in one non-diagonal element triggers the other to be changed as well.

\(^{52}\)it is actually one small number, \(h\), that is multiplied with the current stress value so that the relative order of magnitude is retained.

---
functions that number should be larger. Here *macheps* is the so called *machine epsilon*. It is defined as
the smallest distinguishable positive number, such that \( 1.0 + macheps > 1.0 \) on the given platform. For example, on the Intel x86 platform \( macheps = 1.08E - 19 \) while on the SUN Sparc and DEC platforms \( macheps = 2.22E - 16 \). It has been found that in the case of yield or potential functions the
best approximation of analytical gradients is obtained by using \( h = \sqrt{macheps} 10^3 \). The three order of
magnitude increase in the finite difference step is due to a rather complicated formula for yield and
potential functions. The error in the approximation, \( \text{approx.} F_{ij} \) is found to be after the \( N^{th} \) decimal
place, where \( N \) is the order of \( macheps \), i.e. \( macheps = O(N) \).

Second derivative approximations for one variable function are given in the form:

\[
a = \frac{(f(x + h_ie_i + h_je_j) - f(x + h_ie_i)) - (f(x + h_je_j) - f(x))}{h_ih_j}
\]  

(104.153)

If the first derivatives are available in closed form, one could use equations (104.151) and (104.152) just
by replacing the function values with tensor values for analytical derivatives.

However, if the analytic derivatives are not available, one has to devise a formula that will create a
fourth order tensor from the changes in two dimensional stress tensors, \( \sigma_{ij} \) and \( \sigma_{kl} \). Using the scheme
employed in equation (104.153) the following scheme has been devised:

\[
\text{approx.} Q_{ijkl} = \frac{(Q(\sigma_{mn} + h_{ij} + h_{kl}) - Q(\sigma_{mn} + h_{ij})) - (Q(\sigma_{mn} + h_{kl}) - Q(\sigma_{mn}))}{h_{ij}h_{kl}}
\]  

(104.154)

Special considerations are necessary in order to retain symmetry of the fourth order tensor. At
the moment it has not been possible to figure out how to build the finite difference approximation to
the second derivatives of yield/potential functions for a general stress state. The only finite difference
approximation of the second derivatives that appears to have worked was the one devised in principal
stress space. Namely, diagonal elements of the analytical and the approximate gradients matched exactly,
but development of non-diagonal elements, and the whole scheme of symmetrizing the fourth order
approximation, still remain a mystery. However, some pattern was observed in non–diagonal elements,
and the work on symmetrizing it is in progress.

---

53In a given precision, i.e. float \( \text{real*4} \), double \( \text{real*8} \) or long double \( \text{real*10} \).
54The precision sought was double \( \text{real*8} \).
55PC computers.
56One should not forget that we work with six dimensional tensor formulae directly.
57see Dennis and Schnabel (1983), section 5.6.
For many different potential functions (or yield functions) the only task left would be the derivation of the first derivatives of $F$ and $Q$ and the second derivatives of $Q$ with respect to $p$, $q$ and $\theta$, namely the first derivatives $\frac{\partial Q}{\partial p}$, $\frac{\partial Q}{\partial q}$ and $\frac{\partial Q}{\partial \theta}$ and the second derivatives $\frac{\partial^2 Q}{\partial p^2}$, $\frac{\partial^2 Q}{\partial p \partial q}$, $\frac{\partial^2 Q}{\partial p \partial \theta}$, $\frac{\partial^2 Q}{\partial q \partial p}$, $\frac{\partial^2 Q}{\partial q^2}$, $\frac{\partial^2 Q}{\partial q \partial \theta}$, $\frac{\partial^2 Q}{\partial \theta \partial p}$, $\frac{\partial^2 Q}{\partial \theta \partial q}$ and $\frac{\partial^2 Q}{\partial \theta^2}$. If the potential function is twice differentiable with respect to the stress tensor $\sigma_{ij}$, and if it is continuous then the Hessian matrix is symmetric.

### 104.5 Line Search Technique for Constitutive Elastic-Plastic Integration

This section is entirely based on Jeremić (2001). There exist a repetition of some previously defined equations...

### 104.6 Elastic and Elastic–Plastic Material Models for Solids

In this section we present elements of general elastic and elastic–plastic material models for engineering materials. We describe various forms of the yield functions, plastic flow directions and hardening and softening laws.

#### 104.6.1 Elasticity

**DSL COMMANDS** for the elastic material models are given in section 205.3 on page 791.

In linear elasticity the relationship between the stress tensor $\sigma_{ij}$ and the strain tensor $\epsilon_{kl}$ can be represented in the following form:

$$\sigma_{ij} = \sigma(\epsilon_{ij}) \quad (104.155)$$

If we assume the existence of a strain energy function $W(\epsilon_{ij})$ then the stress strain relation is:

$$\sigma_{ij} = \frac{\partial W(\epsilon_{ij})}{\partial \epsilon_{ij}} \quad (104.156)$$

The introduction of the strain energy density function into elasticity is due to Green, and elastic solids for which such a function is assumed to exist are called Green elastic or hyperelastic solids.

---

58 per unit volume.
Linearization of an elastic continuum is carried out with respect to a reference configuration which is stress free at temperature $T_0$, so that $\sigma_{ij} = 0$. If we denote as $E_{ijkl}$ an isothermal modulus tensor, then under isothermal conditions, we obtain the generalized Hooke’s law:

$$\sigma_{ij} = E_{ijkl} \epsilon_{kl}$$

(104.157)

where $E_{ijkl}$ is the fourth order elastic stiffness tensor with 81 independent components in total. The elastic stiffness tensor features both minor symmetry $E_{ijkl} = E_{jikl} = E_{ijlk}$ and major symmetry $E_{ijkl} = E_{klij}$ (Jeremić and Sture, 1997). The number of independent components for such elastic stiffness tensor is 21 (Spencer, 1980).

$$E_{ijkl} = \left| \frac{\partial^2 W}{\partial \epsilon_{ij} \partial \epsilon_{kl}} \right|_{\epsilon=0} = \left| \frac{\partial^2 W}{\partial \epsilon_{kl} \partial \epsilon_{ij}} \right|_{\epsilon=0}$$

(104.158)

We will restrain our considerations to the isotropic case. The most general form of the isotropic tensor of rank 4 has the following representation:

$$I_4 = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})$$

(104.159)

If $E_{ijkl}$ has this form then in order to satisfy the symmetry condition\(^{59}\) $E_{ijkl} = E_{jikl}$ we must have $\nu = \mu$. The symmetry condition\(^{60}\) $E_{ijkl} = E_{klij}$ is then automatically satisfied. The elastic constant tensor has the following form:

$$E_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})$$

(104.160)

where $\lambda$ and $\mu$ are the Lamé coefficients:

$$\lambda = \frac{\nu E}{(1 + \nu)(1 - 2\nu)} ; \quad \mu = \frac{E}{2(1 + \nu)}$$

(104.161)

and $E$ and $\nu$ are Young’s Modulus and Poisson’s ratio respectively. The symmetric part of the fourth order unit tensor is:

$$I_{ijkl}^{\text{sym}} = \frac{1}{2} (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})$$

(104.162)

\(^{59}\) symmetry in stress tensor.

\(^{60}\) existence of strain energy function.
and can be found as multiplier of $\mu$ in equation (104.160). Equation (104.160) can be written in terms of $E$ and $\nu$ as:

$$E_{ijkl} = \frac{E}{2(1 + \nu)} \left( \frac{2\nu}{1 - 2\nu} \delta_{ij} \delta_{kl} + \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} \right)$$  \hfill (104.163)

The same relation in terms of bulk modulus $K$ and shear modulus $G$ is:

$$E_{ijkl} = K \delta_{ij} \delta_{kl} + G \left( -\frac{2}{3} \delta_{ij} \delta_{kl} + \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} \right)$$  \hfill (104.164)

where $K$ and $G$ are given as:

$$K = \lambda + \frac{2}{3} \mu ; \quad G = \mu$$  \hfill (104.165)

The relation between the strain tensor, $\epsilon_{kl}$ and the stress tensor, $\sigma_{ij}$ is:

$$\epsilon_{kl} = D_{klpq} \sigma_{pq}$$  \hfill (104.166)

where $D_{klpq}$ is the elastic compliance fourth order tensor, defined as:

$$D_{klpq} = -\frac{\lambda}{2\mu (3\lambda + 2\mu)} \delta_{kl} \delta_{pq} + \frac{1}{4\mu} (\delta_{kp} \delta_{lq} + \delta_{kq} \delta_{lp})$$  \hfill (104.167)

or in terms of $E$ and $\nu$:

$$D_{klpq} = \frac{1 + \nu}{2E} \left( \frac{-2\nu}{1 + \nu} \delta_{kl} \delta_{pq} + \delta_{kp} \delta_{lq} + \delta_{kq} \delta_{lp} \right)$$  \hfill (104.168)

of in terms of $K$ and $G$:

$$D_{klpq} = \frac{1}{9K} (\delta_{kl} \delta_{pq}) + \frac{1}{2G} \left( -\frac{1}{3} \delta_{kl} \delta_{pq} + \frac{1}{2} (\delta_{kp} \delta_{lq} + \delta_{kq} \delta_{lp}) \right)$$  \hfill (104.169)

It is worthwhile noting that the part adjacent to the inverse of the bulk modulus $K$:

$$(\delta_{kl} \delta_{pq})$$

controls the volumetric response and that the part adjacent to the inverse of the shear modulus $G$:

$$\left( -\frac{1}{3} \delta_{kl} \delta_{pq} + \frac{1}{2} (\delta_{kp} \delta_{lq} + \delta_{kq} \delta_{lp}) \right)$$
controls the shear response! This note will prove useful later on. Linear transformation of the stress
tensor $\sigma_{pq}$ into itself, i.e. $\sigma_{ij}$ is defined as:

$$\sigma_{ij} = E_{ijkl}\epsilon_{kl} = E_{ijkl}D_{kpq}\sigma_{pq}$$  \hspace{1cm} (104.170)

where

$$E_{ijkl}D_{kpq} = \frac{1}{2}(\delta_{ip}\delta_{jq} + \delta_{iq}\delta_{jp}) = I_{ijpq}^{sym}$$  \hspace{1cm} (104.171)

### 104.6.1.1 Elastic Model

Linear elastic law is the simplest one and assumes constant Young’s modulus $E$ and constant Poisson’s
Ration $\nu$.

### 104.6.1.2 Non–linear Elastic Model #1

This nonlinear model (Janbu, 1963), (Duncan and Chang, 1970) assumes dependence of the Young’s
modulus on the minor principal stress $\sigma_3 = \sigma_{\text{min}}$ in the form

$$E = Kp_a\left(\frac{\sigma_3}{p_a}\right)^n$$  \hspace{1cm} (104.172)

Here, $p_a$ is the atmospheric pressure in the same units as $E$ and stress. The two material constants $K$
and $n$ are constant for a given void ratio.

### 104.6.1.3 Non–linear Elastic Model #2

If Young’s modulus and Poisson’s ratio are replaced by the shear modulus $G$ and bulk modulus $K$ the
non–linear elastic relationship can be expressed in terms of the normal effective mean stress $p$ as

$$G \quad \text{and/or} \quad K = AF(e,OCR)p^n$$  \hspace{1cm} (104.173)

where $e$ is the void ratio, $OCR$ is the overconsolidation ratio and $p = \sigma_{ii}/3$ is the mean effective stress
(Hardin, 1978).

### 104.6.1.4 Lade’s Non–linear Elastic Model

Lade and Nelson (1987) and Lade (1988a) proposed a nonlinear elastic model based on Hooke’s law in
which Poisson ratio $\nu$ is kept constant. According to this model, Young’s modulus can be expressed in
terms of a power law as:

\[ E = M p_a \left( \left( \frac{I_1}{p_a} \right)^2 + \left( \frac{6 + \nu}{1 - 2\nu} \frac{J_{2D}}{p_a^2} \right)^\lambda \right) \]  

(104.174)

where \( I_1 = \sigma_{ii} \) is the first invariant of the stress tensor and \( J_{2D} = (s_{ij}s_{ij})/2 \) is the second invariant of the deviatoric stress tensor \( s_{ij} = \sigma_{ij} - \sigma_{kk}\delta_{ij}/3 \). The parameter \( p_a \) is atmospheric pressure expressed in the same unit as \( E, I_1 \) and \( \sqrt{J_{2D}} \) and the modulus number \( M \) and the exponent \( \lambda \) are constant, dimensionless numbers.

### 104.6.1.5 Cross Anisotropic Linear Elastic Model

#### 104.6.2 Yield Functions

The typical plastic behavior of frictional materials is influenced by both normal and shear stresses. It is usually assumed that there exists a yield surface \( F \) in the stress space that encompasses the elastic region. States of stress inside the yield surface are assumed to be elastic (linear or non-linear). Stress states on the surface are assumed to produce plastic deformations. Yield surfaces for geomaterials are usually shaped as asymmetric tar drops with smoothly rounded triangular cross sections. In addition to that, simpler yield surfaces, based on the Drucker–Prager cone or Mohr–Coulomb hexagon can also be successfully used if matched with appropriate hardening laws. Yield surface shown in Figure 104.6 Lade (1988b) represent typical meridian plane trace for an isotropic granular material. Line BC represents stress path for conventional triaxial compression test. Figure 104.7 represents the view of the yield surface traces in the deviatoric plane.

#### 104.6.3 Plastic Flow Directions

Plastic flow directions are traditionally derived from a potential surface which to some extent reassembles the yield surface. Potential surfaces for metals are the same as their yield surfaces but experimental evidence suggests that it is not the case for geomaterials. The non–associated flow rules, used in geomechanics, rely on the potential surface, which is different from the yield surface, to provide the plastic flow directions. It should be noted that the potential surface is used for convenience and there is no physical reason to assume that the plastic strain rates are related to a potential surface \( Q \) (Vardoulakis and Sulem, 1995). Instead of defining a plastic potential, one may assume that the plastic flow direction is derived from an tensor function which does not have to possess a potential function.
104.6.4 Hardening–Softening Evolution Laws

The change in size and/or shape of the yield and potential surfaces is controlled by the hardening–softening evolution laws. Physically, these laws control the hardening and/or softening process during loading. Depending on the evolution type they control, these laws can be in general separated into isotropic and kinematic (also called anisotropic). The isotropic evolution laws control the size of the yield surface through a single scalar variable. This is usually related to the Coulomb friction or to the mean stress values at isotropic yielding. The non–isotropic evolution laws can be further specialized to rotational, translational kinematic and distortional. It should be noted that all of the kinematic evolution laws can be treated as special case of the general, distortional laws (Baltov and Sawczuk, 1965). Figure 104.8 depicts various types of evolution laws (for the control of hardening–softening) in the meridian plane.

104.6.5 Tresca Model

The first yield criteria in the metal plasticity is Tresca yield criteria. Tresca yield criteria states that when the maximum shear stress or, the half difference of the maximum and minimum principal stresses, reaches the shear strength, \( \tau_s \), the material will begin yielding. It is can be expressed by the yield function

\[ f(\sigma_1 - \frac{\sigma_3}{2}) = \tau_s \]

\[^{61}\text{The meridian plane is chosen just for illustration purposes, similar sketch can be produced in deviatoric plane as well.}\]
Figure 104.7: Deviatoric trace of typical yield surface for pressure sensitive materials.

\[ f = |\tau_{max}| - \tau_s = \frac{1}{2} |\sigma_1 - \sigma_3| - \tau_s = 0 \]  

(104.175)

Tresca yield surface in the principal stress space is a regular hexagonal cylinder. It is implied that the intermediate principal stress plays no role in the yielding for Tresca yield criteria.

104.6.6 von Mises Model

DSL COMMANDS for the von Mises material models are given in section 205.3 on page 791.

Experimental data showed that for most metals, von Mises yield criteria is more accurate than Tresca criteria. von Mises yield function can be expressed by

\[ f = 3J_2 - k^2 = 0 \]  

(104.176)
Figure 104.8: Various types of evolution laws that control hardening and/or softening of elastic–plastic material models: (a) Isotropic (scalar) controlling equivalent friction angle and isotropic yield stress. (b) Rotational kinematic hardening (second order tensor) controlling pivoting around fixed point (usually stress origin) of the yield surface. (c) Translational kinematic hardening (second order tensor) controlling translation of the yield surface. (d) Distortional (fourth order tensor) controlling the shape of the yield surface.

or if extended to include the kinematic hardening,

\[
f = \frac{3}{2} (s_{ij} - \alpha_{ij}) (s_{ij} - \alpha_{ij}) - k^2 = 0
\]  

(104.177)

where \( k \) is the scalar internal variable; its initial value is the uniaxial tension strength. \( \alpha_{ij} \) is the tensor internal variable called the back stress. Similar to \( s_{ij} \), \( \alpha_{ij} \) is also a deviatoric symmetric tensor.

Although von Mises model is mainly for the metal plasticity analysis, for undrained analysis in geomechanics, von Mises model can be approximately used to simulate the undrained behaviors, (Yang and Jeremić, 2002), (Yang and Jeremić, 2003).

The stress derivative of the yield function is

\[
\frac{\partial f}{\partial \sigma_{ij}} = 3(s_{ij} - \alpha_{ij})
\]  

(104.178)

From Equation 104.178, it is easily to derive that

\[
\frac{\partial f}{\partial \alpha_{ij}} = -3(s_{ij} - \alpha_{ij})
\]  

(104.179)

and

\[
\frac{\partial f}{\partial k} = -2k
\]  

(104.180)
If the associated plastic flow rule \( g = f \) is assumed, then
\[
m_{ij} = \frac{\partial g}{\partial \sigma_{ij}} = 3(s_{ij} - \alpha_{ij}) \tag{104.181}
\]

\[
\frac{\partial m_{ij}}{\partial \sigma_{mn}} = 3I_{ijmn}^s - \delta_{ij}\delta_{mn} \tag{104.182}
\]

\[
\frac{\partial m_{ij}}{\partial \alpha_{mn}} = -3I_{ijmn}^s \tag{104.183}
\]

where \( I_{ijmn}^s \) is the symmetric unit rank-4 tensor.

It is interesting that from the Equation 104.181, von Mises model gives
\[
\dot{\epsilon}_p^v = \dot{\epsilon}_p^{ii} = \dot{\lambda}m_{ii} = 3\dot{\lambda}(s_{ii} - \alpha_{ii}) = 0 \tag{104.184}
\]

which accords with the phenomena that no plastic volumetric strain occurs for metals. It is implied that the isotropic stress (hydrostatic pressure) can never make the metal yield for this yield criteria. von Mises model is therefore pressure-independent.

If \( k \) is assumed a linear relation to the equivalent plastic strain \( \epsilon_p^v \), or by the equation
\[
\dot{k} = H_s \epsilon_p^v = \dot{\lambda}m_{ii} = \frac{2}{3}m_{ij}^{dev}m_{ij}^{dev} \tag{104.185}
\]

where \( H_s \) is the linear hardening/softening modulus to the equivalent plastic strain, the corresponding \( h_A \) is then
\[
h = H_s \left( \frac{2}{3}m_{ij}^{dev}m_{ij}^{dev} \right)^{0.5} \tag{104.186}
\]

where \( m_{ij}^{dev} \) is the ‘deviatoric’ plastic flow, and if it is associated plasticity,
\[
h = 2H_s k \tag{104.187}
\]

If \( \alpha_{ij} \) is assumed a linear relation to the plastic strain tensor \( \epsilon_{ij}^p \), or by the equation
\[
\dot{\alpha}_{ij} = H_t \dot{\epsilon}_{ij}^p = \dot{\lambda}H_t m_{ij} \tag{104.188}
\]

if it is associated plasticity,
\[
\dot{\alpha}_{ij} = 3\dot{\lambda}H_t(s_{ij} - \alpha_{ij}) \tag{104.189}
\]
where $H_t$ is the linear hardening/softening modulus to plastic strain tensor, the corresponding $h_A$ is then

$$h_{ij} = H_t m_{ij} \quad (104.190)$$

if it is associated plasticity,

$$h_{ij} = 3H_t(s_{ij} - \alpha_{ij}) \quad (104.191)$$

A saturation-type kinematic hardening rule is the Armstrong-Frederick hardening (Armstrong and Frederick, 1966),

$$\dot{\alpha}_{ij} = \frac{2}{3} h_a \dot{e}_{ij}^p - c_r \dot{e}_{eq} \alpha_{ij} \quad (104.192)$$

if it is associated plasticity,

$$\dot{\alpha}_{ij} = \dot{\lambda} [h_a (s_{ij} - \alpha_{ij}) - 2c_r k \alpha_{ij}] \quad (104.193)$$

where $h_a$ and $c_r$ are material constants. The corresponding $h_A$ is then

$$h_{ij} = \frac{2}{3} h_a m_{ij} - c_r m_{eq} \alpha_{ij} \quad (104.194)$$

where $m_{eq}$ is the ‘equivalent’ plastic flow, and if it is associated plasticity,

$$h_{ij} = 2h_a s_{ij} - 2(h_a + c_r k) \alpha_{ij} \quad (104.195)$$

### 104.6.6.1 Yield and Plastic Potential Functions: von Mises Model (form I)

**Yield function and related derivatives**

$$f = \frac{3}{2} [(s_{ij} - \alpha_{ij})(s_{ij} - \alpha_{ij})] - k^2 = 0 \quad (104.196)$$

$$\frac{\partial f}{\partial \sigma_{ij}} = 3 \frac{\partial s_{kl}}{\partial \sigma_{ij}}(s_{kl} - \alpha_{kl})$$

$$= 3 (\delta_{kl} \delta_{ij} - \frac{1}{3} \delta_{kl} \delta_{ij}) (s_{kl} - \alpha_{kl})$$

$$= 3 (s_{ij} - \alpha_{ij}) \quad (104.197)$$

$$\frac{\partial f}{\partial \alpha_{ij}} = -3 \frac{\partial s_{kl}}{\partial \alpha_{ij}}(s_{kl} - \alpha_{kl})$$

$$= -3 \delta_{kl} \delta_{ij} (s_{kl} - \alpha_{kl})$$

$$= -3 (s_{ij} - \alpha_{ij}) \quad (104.198)$$
\[
\frac{\partial f}{\partial k} = -2k \quad (104.199)
\]

**Plastic flow (associated plasticity) and related derivatives**

\[
m_{ij} = \frac{\partial f}{\partial \sigma_{ij}} = 3 (s_{ij} - \alpha_{ij}) \quad (104.200)
\]

\[
\frac{\partial m_{ij}}{\partial \sigma_{mn}} = 3 \delta_{im} \delta_{jn} - \delta_{ij} \delta_{mn} \quad (104.201)
\]

\[
\frac{\partial m_{ij}}{\partial k} = 0 \quad (104.202)
\]

\[
\frac{\partial m_{ij}}{\partial \alpha_{mn}} = -3 \delta_{im} \delta_{jn} \quad (104.203)
\]

**104.6.6.2 Yield and Plastic Potential Functions: von Mises Model (form II)**

**Yield function and related derivatives**

\[
f = [(s_{ij} - \alpha_{ij}) (s_{ij} - \alpha_{ij})]^{0.5} - \sqrt{\frac{2}{3}} k = 0 \quad (104.204)
\]

\[
\frac{\partial f}{\partial \sigma_{ij}} = \frac{\partial s_{kl}}{\partial \sigma_{ij}} (s_{kl} - \alpha_{kl}) [(s_{mn} - \alpha_{mn}) (s_{mn} - \alpha_{mn})]^{-0.5}
\]

\[
= \left[ \delta_{ki} \delta_{lj} - \frac{1}{3} \delta_{kl} \delta_{ij} \right] (s_{kl} - \alpha_{kl}) [(s_{mn} - \alpha_{mn}) (s_{mn} - \alpha_{mn})]^{-0.5}
\]

\[
= (s_{ij} - \alpha_{ij}) [(s_{mn} - \alpha_{mn}) (s_{mn} - \alpha_{mn})]^{-0.5} \quad (104.205)
\]

\[
\frac{\partial f}{\partial \alpha_{ij}} = - (s_{ij} - \alpha_{ij}) [(s_{mn} - \alpha_{mn}) (s_{mn} - \alpha_{mn})]^{-0.5} \quad (104.206)
\]

\[
\frac{\partial f}{\partial k} = - \sqrt{\frac{2}{3}} \quad (104.207)
\]
Plastic flow (associated plasticity) and related derivatives

\[
m_{ij} = \frac{\partial f}{\partial \sigma_{ij}} = (s_{ij} - \alpha_{ij}) \left[ (s_{mn} - \alpha_{mn}) (s_{mn} - \alpha_{mn}) \right]^{-0.5}
\]  
(104.208)

\[
\frac{\partial m_{ij}}{\partial \sigma_{mn}} = \left( \delta_{im} \delta_{jn} - \frac{1}{3} \delta_{ij} \delta_{mn} \right) \left[ (s_{rs} - \alpha_{rs}) (s_{rs} - \alpha_{rs}) \right]^{-0.5} - (s_{ij} - \alpha_{ij}) (s_{mn} - \alpha_{mn}) \left[ (s_{rs} - \alpha_{rs}) (s_{rs} - \alpha_{rs}) \right]^{-1.5}
\]  
(104.209)

\[
\frac{\partial m_{ij}}{\partial \sigma_{ij}} = 0
\]  
(104.210)

\[
\frac{\partial m_{ij}}{\partial \alpha_{mn}} = -\delta_{im} \delta_{jn} \left[ (s_{rs} - \alpha_{rs}) (s_{rs} - \alpha_{rs}) \right]^{-0.5} + (s_{ij} - \alpha_{ij}) (s_{mn} - \alpha_{mn}) \left[ (s_{rs} - \alpha_{rs}) (s_{rs} - \alpha_{rs}) \right]^{-1.5}
\]  
(104.211)

### 104.6.6.3 Hardening and Softening Functions: von Mises Model

**Linear isotropic hardening and related derivatives**

\[
\tilde{k} = Hm_{equivalent} = H \left( \frac{2}{3} m_{ij}m_{ij} \right)^{0.5}
\]  
(104.212)

\[
\frac{\partial \tilde{k}}{\partial \sigma_{ij}} = \frac{2}{3} Hm_{pq} \frac{\partial m_{pq}}{\partial \sigma_{ij}} \left( \frac{2}{3} m_{mn}m_{mn} \right)^{-0.5}
\]  
(104.213)

\[
\frac{\partial \tilde{k}}{\partial k} = \frac{2}{3} Hm_{pq} \frac{\partial m_{pq}}{\partial \sigma_{ij}} \left( \frac{2}{3} m_{mn}m_{mn} \right)^{-0.5}
\]  
(104.214)

\[
\frac{\partial \tilde{k}}{\partial \alpha_{ij}} = \frac{2}{3} Hm_{pq} \frac{\partial m_{pq}}{\partial \sigma_{ij}} \left( \frac{2}{3} m_{mn}m_{mn} \right)^{-0.5}
\]  
(104.215)

**Linear kinematic hardening and related derivatives**

\[
\tilde{\alpha}_{ij} = Hm_{ij}^{dev} = H \left( m_{ij} - \frac{1}{3} m_{kl} \delta_{kl} \delta_{ij} \right)
\]  
(104.216)

\[
\frac{\partial \tilde{\alpha}_{ij}}{\partial \sigma_{ij}} = H \left( \frac{\partial m_{ij}}{\partial \sigma_{mn}} - \frac{1}{3} \frac{\partial m_{kl}}{\partial \sigma_{mn}} \delta_{kl} \delta_{ij} \right)
\]  
(104.217)
\[
\frac{\partial \bar{\alpha}_{ij}}{\partial k} = H \left( \frac{\partial m_{ij}}{\partial k} - \frac{1}{3} \frac{\partial m_{kl}}{\partial k} \delta_{kl} \delta_{ij} \right) \tag{104.218}
\]

\[
\frac{\partial \bar{\alpha}_{ij}}{\partial \alpha_{mn}} = H \left( \frac{\partial m_{ij}}{\partial \alpha_{mn}} - \frac{1}{3} \frac{\partial m_{kl}}{\partial \alpha_{mn}} \delta_{kl} \delta_{ij} \right) \tag{104.219}
\]

**Armstrong-Frederick kinematic hardening for von Mises**

\[
\bar{\alpha}_{ij} = \frac{2}{3} h_a m_{ij}^{dev} - c_r \left( \frac{2}{3} m_{rs}^{dev} m_{rs}^{dev} \right)^{0.5} \alpha_{ij} \tag{104.220}
\]

The unit of parameter \( h_a \) is Pascal. The parameter \( c_r \) is unitless. The unit of \( \alpha_{ij} \) is Pascal.

The deviatoric component of \( m \) is employed because the backstress \( \bar{\alpha}_{ij} \) is the center of yield surface in the deviatoric stress space.

When the derivative of backstress \( \bar{\alpha}_{ij} = 0 \), the tensor \( \alpha_{ij} \) reaches the tensor limit.

\[
\alpha_{ij}^{lim} = \sqrt{\frac{2}{3}} \frac{h_a m_{ij}^{dev}}{c_r \sqrt{m_{rs}^{dev} m_{rs}^{dev}}} \tag{104.221}
\]

Some useful tensor derivatives for von Mises \( \bar{\alpha} \).

\[
m_{ij}^{dev} = m_{ij} - \frac{1}{3} m_{kl} \delta_{kl} \delta_{ij} \tag{104.222}
\]

- Useful tensor derivatives for von Mises \( \bar{\alpha} \) with respect to \( \sigma \).

\[
\frac{\partial \bar{\alpha}_{ij}}{\partial \sigma_{mn}} = \frac{2}{3} \frac{h_a m_{ij}^{dev}}{\sigma_{mn}} - \frac{2}{3} c_r m_{rs}^{dev} \frac{\partial m_{ij}^{dev}}{\partial \sigma_{mn}} \left( \frac{2}{3} m_{kl}^{dev} m_{kl}^{dev} \right)^{-0.5} \alpha_{ij} \tag{104.223}
\]

where

\[
\frac{\partial m_{ij}^{dev}}{\partial \sigma_{mn}} = \frac{\partial m_{ij}}{\partial \sigma_{mn}} - \frac{1}{3} \frac{\partial m_{ot}}{\partial \sigma_{mn}} \delta_{ot} \delta_{ij} \tag{104.224}
\]

- Useful tensor derivatives for von Mises \( \bar{\alpha} \) with respect to \( \alpha \).

\[
\frac{\partial \bar{\alpha}_{ij}}{\partial \alpha_{mn}} = \frac{2}{3} \frac{h_a m_{ij}^{dev}}{\alpha_{mn}} - \frac{2}{3} c_r m_{rs}^{dev} \frac{\partial m_{ij}^{dev}}{\partial \alpha_{mn}} \left( \frac{2}{3} m_{kl}^{dev} m_{kl}^{dev} \right)^{-0.5} \alpha_{ij} - c_r \left( \frac{2}{3} m_{pq}^{dev} m_{pq}^{dev} \right)^{0.5} \delta_{im} \delta_{jn} \tag{104.225}
\]

where

\[
\frac{\partial m_{ij}^{dev}}{\partial \alpha_{mn}} = \frac{\partial m_{ij}}{\partial \alpha_{mn}} - \frac{1}{3} \frac{\partial m_{ot}}{\partial \alpha_{mn}} \delta_{ot} \delta_{ij} \tag{104.226}
\]
Useful tensor derivatives for von Mises $\bar{\alpha}$ with respect to $k$.

$$\frac{\partial \bar{\alpha}_{ij}}{\partial k} = \frac{2}{3} h_a \frac{\partial m_{ij}^{dev}}{\partial k} - \sqrt{\frac{2}{3}} c_r \alpha_{ij} (m_{rs}^{dev} m_{rs}^{dev})^{-0.5} \frac{\partial m_{pq}^{dev}}{\partial k} m_{pq}^{dev}$$  (104.227)

where

$$\frac{\partial m_{ij}^{dev}}{\partial k} = \frac{\partial m_{ij}}{\partial k} - \frac{1}{3} \frac{\partial m_{ot}}{\partial k} \delta_{ot} \delta_{ij}$$  (104.228)

### 104.6.7 Drucker-Prager Model

**DSL COMMANDS** for the Drucker Prager material models are given in section 205.3 on page 791.

Drucker and Prager (1952) proposed a right circle cone to match with the Mohr-Coulomb irregular hexagonal pyramid, which can be expressed by

$$f = \alpha I_1 + \sqrt{J_2} - \beta = 0$$  (104.229)

or if considering the kinematic hardening,

$$f = \alpha I_1 + \left[\frac{1}{2} (s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})\right]^{\frac{1}{2}} - \beta = 0$$  (104.230)

where $\alpha$ and $\beta$ are material constants.

By coinciding Drucker-Prager cone with the outer apexes of the Mohr-Coulomb hexagon locus, we get the compressive cone of Drucker-Prager model, with the constants as

$$\alpha = \frac{2 \sin \phi}{\sqrt{3}(3 - \sin \phi)} \cdot \beta = \frac{6 \cos \phi}{\sqrt{3}(3 - \sin \phi)} c$$  (104.231)

By coinciding Drucker-Prager cone with the inner apexes of the Mohr-Coulomb hexagon locus, we get the tensile cone of Drucker-Prager model, with the constants as

$$\alpha = \frac{2 \sin \phi}{\sqrt{3}(3 + \sin \phi)} \cdot \beta = \frac{6 \cos \phi}{\sqrt{3}(3 + \sin \phi)} c$$  (104.232)

We can also get the mean cone of the compressive and tensile cone, with the constants as

$$\alpha = \frac{\sqrt{3} \sin \phi}{9 - \sin^2 \phi} \cdot \beta = \frac{2\sqrt{3} \cos \phi}{9 - \sin^2 \phi} c$$  (104.233)

Another inner-tangent cone to the the Mohr-Coulomb pyramid, with the constants as

$$\alpha = \frac{\tan \phi}{\sqrt{9 + 12 \tan^2 \phi}} \cdot \beta = \frac{3c}{\sqrt{9 + 12 \tan^2 \phi}}$$  (104.234)

Obviously, in practice $\alpha$ and $\beta$ are not directly obtained from experiments. They are functions of Mohr-Coulomb parameters, the cohesion $c$ and the friction angle $\phi$, which can be determined by
experiments. The shape of Drucker-Prager yield surface has different types. They only partially satisfy the above requirements for locus in the $\pi$ plane: they do not coincide with both compressive and tensile experimental points.

A useful formulation on Equation 104.229 is

$$\frac{\partial f}{\partial \sigma_{ij}} = \alpha \delta_{ij} + \frac{s_{ij}}{2\sqrt{J_2}}$$

(104.235)

For cohesionless sands, $k = 0$, Drucker-Prager yield function can thus be simplified as

$$f = \alpha I_1 + \sqrt{J_2} = 0$$

(104.236)

or in terms of $p$ and $q$,

$$f = q - Mp = 0$$

(104.237)

If Equation 104.231 is adopted, then $M$ can be easily derived as

$$M = \frac{6 \sin \phi}{3 - \sin \phi}$$

(104.238)

If the kinematic hardening is taken account, Equation 104.237 can be extended into

$$f = \frac{3}{2}[(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})] - M^2p^2 = 0$$

(104.239)

Useful formulations for this yield function are

$$\frac{\partial f}{\partial \sigma_{ij}} = 3\bar{s}_{ij} + \left(\bar{s}_{mn}\alpha_{mn} + \frac{2}{3}M^2p\right)\delta_{ij}$$

(104.240)

$$\frac{\partial f}{\partial \alpha_{ij}} = -3p\bar{s}_{ij}$$

(104.241)

where $\bar{s}_{ij} = s_{ij} - p\alpha_{ij}$.

If the plastic flow is assumed associated, $g = f$, then

$$m_{ij} = \frac{\partial g}{\partial \sigma_{ij}} = 3\bar{s}_{ij} + \left(\bar{s}_{mn}\alpha_{mn} + \frac{2}{3}M^2p\right)\delta_{ij}$$

(104.242)

the 'deviatoric' plastic flow is therefore

$$m_{eq} = 2Mp$$

(104.243)
104.6.7.1 Yield and Plastic Potential Functions: Drucker-Prager Model (form I)

Yield function and related derivatives

\[ f = \frac{3}{2} [(s_{ij} - p\alpha_{ij}) (s_{ij} - p\alpha_{ij})] - k^2 p^2 = 0 \] (104.244)

\[ \frac{\partial f}{\partial \sigma_{ij}} = \frac{3}{2} \left[ 2 \frac{\partial s_{mn}}{\partial \sigma_{ij}} (s_{mn} - p\alpha_{mn}) \right] + \frac{3}{2} \left[ -2\alpha_{mn} \frac{\partial p}{\partial \sigma_{ij}} (s_{mn} - p\alpha_{mn}) \right] - 2k^2 p \frac{\partial p}{\partial \sigma_{ij}} \]

\[ = 3 \left( \delta_{mi} \delta_{nj} - \frac{1}{3} \delta_{mn} \delta_{ij} \right) (s_{mn} - p\alpha_{mn}) + 3 \left[ \alpha_{mn} \frac{1}{3} \delta_{ij} (s_{mn} - p\alpha_{mn}) \right] + \frac{2}{3} k^2 p \delta_{ij} \]

\[ = 3 (s_{ij} - p\alpha_{ij}) + \alpha_{mn} (s_{mn} - p\alpha_{mn}) \delta_{ij} + \frac{2}{3} k^2 p \delta_{ij} \] (104.245)

\[ \frac{\partial f}{\partial \alpha_{ij}} = -3p (s_{ij} - p\alpha_{ij}) \] (104.246)

\[ \frac{\partial f}{\partial k} = -2k p^2 \] (104.247)

Plastic flow (associated plasticity) and related derivatives

\[ m_{ij} = \frac{\partial f}{\partial \sigma_{ij}} = 3 (s_{ij} - p\alpha_{ij}) + \alpha_{rs} (s_{rs} - p\alpha_{rs}) \delta_{ij} + \frac{2}{3} k^2 p \delta_{ij} \] (104.248)

\[ \frac{\partial m_{ij}}{\partial \sigma_{mn}} = 3 \left( \left( \delta_{im} \delta_{jn} - \frac{1}{3} \delta_{ij} \delta_{mn} \right) - \frac{1}{3} \delta_{mn} \alpha_{ij} \right) + \alpha_{rs} \frac{\partial (s_{rs} - p\alpha_{rs})}{\partial \sigma_{mn}} \delta_{ij} + \frac{2}{3} k^2 p \frac{\partial p}{\partial \sigma_{mn}} \delta_{ij} \]

\[ = 3\delta_{im} \delta_{jn} - \delta_{ij} \delta_{mn} - \delta_{mn} \alpha_{ij} + \alpha_{rs} \left( \delta_{rm} \delta_{sn} - \frac{1}{3} \delta_{rs} \delta_{mn} + \frac{1}{3} \delta_{mn} \alpha_{rs} \right) \delta_{ij} + \frac{2}{3} k^2 p \delta_{ij} \]

\[ = 3\delta_{im} \delta_{jn} - \delta_{ij} \delta_{mn} - \delta_{mn} \alpha_{ij} + \alpha_{mn} \delta_{ij} + \frac{1}{3} \delta_{mn} \alpha_{rs} \alpha_{rs} \delta_{ij} - \frac{2}{9} k^2 \delta_{mn} \delta_{ij} \]

\[ = 3\delta_{im} \delta_{jn} + \left( -1 + \frac{1}{3} \alpha_{rs} \alpha_{rs} - \frac{2}{9} k^2 \right) \delta_{ij} \delta_{mn} - \delta_{mn} \alpha_{ij} + \alpha_{mn} \delta_{ij} \] (104.249)

\[ \frac{\partial m_{ij}}{\partial k} = \frac{4}{3} kp \delta_{ij} \] (104.250)

\[ \frac{\partial m_{ij}}{\partial \alpha_{mn}} = -3p \delta_{im} \delta_{jn} + \delta_{rm} \delta_{sn} (s_{rs} - p\alpha_{rs}) \delta_{ij} - \alpha_{rs} p \delta_{rm} \delta_{sn} \delta_{ij} \]

\[ = -3p \delta_{im} \delta_{jn} + (s_{mn} - p\alpha_{mn}) \delta_{ij} - \alpha_{mn} p \delta_{ij} \]

\[ = -3p \delta_{im} \delta_{jn} + s_{mn} \delta_{ij} - 2p \alpha_{mn} \delta_{ij} \] (104.251)
104.6.7.2 Yield and Plastic Potential Functions: Drucker-Prager Model (form II)

Yield function and related derivatives

\[ f = [(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})]^{0.5} - \sqrt{\frac{2}{3}}kp = 0 \] (104.252)

\[
\frac{\partial f}{\partial \sigma_{ij}} = \left( \frac{\partial s_{mn}}{\partial \sigma_{ij}} - \alpha_{mn} \frac{\partial p}{\partial \sigma_{ij}} \right) (s_{mn} - p\alpha_{mn}) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} - \sqrt{\frac{2}{3}}k \frac{\partial p}{\partial \sigma_{ij}} \\
= \left( \delta_{mi}\delta_{nj} - \frac{1}{3} \delta_{mn}\delta_{ij} + \frac{1}{3} \alpha_{mn}\delta_{ij} \right) (s_{mn} - p\alpha_{mn}) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} + \sqrt{\frac{2}{27}}k\delta_{ij}
\]

\[
\frac{\partial f}{\partial \alpha_{ij}} = -p\delta_{mi}\delta_{nj} (s_{mn} - p\alpha_{mn}) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} \\
= -p (s_{ij} - p\alpha_{ij}) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} (104.254)
\]

\[
\frac{\partial f}{\partial k} = -\sqrt{\frac{2}{3}}p
\] (104.255)

Plastic flow (associated plasticity) and related derivatives

\[ m_{ij} = \frac{\partial f}{\partial \sigma_{ij}} = \left( (s_{ij} - p\alpha_{ij}) + \frac{1}{3} \alpha_{pq}\delta_{ij} (s_{pq} - p\alpha_{pq}) \right) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} + \sqrt{\frac{2}{27}}k\delta_{ij} \] (104.256)

\[
\frac{\partial m_{ij}}{\partial \sigma_{mn}} = \left[ \left( \delta_{mi}\delta_{nj} - \frac{1}{3} \delta_{mn}\delta_{ij} + \frac{1}{3} \alpha_{mn}\delta_{ij} \right) + \frac{1}{3} \alpha_{pq}\delta_{ij} \left( \delta_{mp}\delta_{nq} - \frac{1}{3} \delta_{mn}\delta_{pq} + \frac{1}{3} \delta_{mn}\alpha_{pq} \right) \right] [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} \]
\[ - \left[ (s_{ij} - p\alpha_{ij}) + \frac{1}{3} \alpha_{pq}\delta_{ij} (s_{pq} - p\alpha_{pq}) \right] \left( \delta_{mp}\delta_{nq} - \frac{1}{3} \delta_{mn}\delta_{pq} + \frac{1}{3} \delta_{mn}\alpha_{pq} \right) \]
\[ (s_{rs} - p\alpha_{rs}) [(s_{tu} - p\alpha_{tu})(s_{tu} - p\alpha_{tu})]^{-1.5} \]
\] (104.257)

\[
\frac{\partial m_{ij}}{\partial k} = \sqrt{\frac{2}{27}}\delta_{ij}
\] (104.258)
\[ \frac{\partial m_{ij}}{\partial \alpha_{mn}} = \left[ -p\delta_{mi}\delta_{nj} + \frac{1}{3}\delta_{mp}\delta_{nq}\delta_{ij}\left(s_{pq} - p\alpha_{pq}\right) - \frac{1}{3}p\alpha_{pq}\delta_{ij}\delta_{mp}\delta_{nq}\right] \left[(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})\right]^{-0.5} \\
- \left[(s_{ij} - p\alpha_{ij}) + \frac{1}{3}\alpha_{pq}\delta_{ij}\left(s_{pq} - p\alpha_{pq}\right)\right] \left[-p\delta_{rm}\delta_{sn}\left(s_{rs} - p\alpha_{rs}\right)\left[(s_{tu} - p\alpha_{tu})\left(s_{tu} - p\alpha_{tu}\right)\right]^{-1.5} \right. \\
\] (104.259)

**Plastic flow (non-associated plasticity) and related derivatives**

\[ m_{ij} = \left( \frac{\partial f}{\partial \sigma_{ij}} \right)^{dev} \left[ s_{ij} - p\alpha_{ij} \right] \left[(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})\right]^{-0.5} - \frac{1}{3}D\delta_{ij} \] (104.260)

where

\[ D = \xi \left( \sqrt{\frac{2}{3}} k_d - \frac{\sqrt{s_{mn}s_{mn}}}{p} \right) \] (104.261)

\[ \frac{\partial D}{\partial \sigma_{mn}} = -p^{-1}s_{mn}(s_{kl}s_{kl})^{-0.5} - \frac{1}{3}p^{-2}\delta_{mn}(s_{ot}s_{ot})^{0.5} \] (104.262)

Therefore,

\[ \frac{\partial m_{ij}}{\partial \sigma_{mn}} = \left( \delta_{mi}\delta_{nj} - \frac{1}{3}\delta_{mn}\delta_{ij} + \frac{1}{3}\delta_{mn}\alpha_{ij}\right) \left[(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})\right]^{-0.5} \\
- \left[(s_{ij} - p\alpha_{ij}) \left(\delta_{mr}\delta_{ns} - \frac{1}{3}\delta_{mn}\delta_{rs} + \frac{1}{3}\delta_{mn}\alpha_{rs}\right)\right] \left[(s_{tu} - p\alpha_{tu})(s_{tu} - p\alpha_{tu})\right]^{-1.5} \\
+ \frac{1}{3}\delta_{ij}s_{mn}p^{-1}(s_{kl}s_{kl})^{-0.5} + \frac{1}{9}\delta_{ij}\delta_{mn}p^{-2}(s_{ot}s_{ot})^{0.5} \] (104.263)

\[ \frac{\partial m_{ij}}{\partial k} = 0 \] (104.264)

\[ \frac{\partial m_{ij}}{\partial \alpha_{mn}} = -p\delta_{mi}\delta_{nj} \left[(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})\right]^{-0.5} \\
- \left[(s_{ij} - p\alpha_{ij}) \left[-p\delta_{rm}\delta_{sn}\left(s_{rs} - p\alpha_{rs}\right)\right] \left[(s_{tu} - p\alpha_{tu})(s_{tu} - p\alpha_{tu})\right]^{-1.5} \right. \] (104.265)

### 104.6.7.3 Hardening and Softening Functions: Drucker-Prager Model

Note that the linear isotropic and linear kinematic hardening equations for Drucker-Prager model are the same as the ones for von Mises model. Here they are shown again for completeness.
Linear isotropic hardening and related derivatives

\[ k = H m^{\text{equivalent}} = H \left( \frac{2}{3} m_{ij} m_{ij} \right)^{0.5} \] (104.266)

\[ \frac{\partial k}{\partial \sigma_{ij}} = \frac{2}{3} H m_{pq} \frac{\partial m_{pq}}{\partial \sigma_{ij}} \left( \frac{2}{3} m_{mn} m_{mn} \right)^{-0.5} \] (104.267)

\[ \frac{\partial k}{\partial \sigma_{ij}} = \frac{2}{3} H m_{pq} \frac{\partial m_{pq}}{\partial \sigma_{ij}} \left( \frac{2}{3} m_{mn} m_{mn} \right)^{-0.5} \] (104.268)

\[ \frac{\partial k}{\partial \sigma_{ij}} = \frac{2}{3} H m_{pq} \frac{\partial m_{pq}}{\partial \sigma_{ij}} \left( \frac{2}{3} m_{mn} m_{mn} \right)^{-0.5} \] (104.269)

Linear kinematic hardening and related derivatives

\[ \bar{\alpha}_{ij} = H m_{ij}^{\text{dev}} = H \left( m_{ij} - \frac{1}{3} m_{kl} \delta_{kl} \delta_{ij} \right) \] (104.270)

\[ \frac{\partial \bar{\alpha}_{ij}}{\partial \sigma_{mn}} = H \left( \frac{\partial m_{ij}}{\partial \sigma_{mn}} - \frac{1}{3} \frac{\partial m_{kl}}{\partial \sigma_{mn}} \delta_{kl} \delta_{ij} \right) \] (104.271)

\[ \frac{\partial \bar{\alpha}_{ij}}{\partial \sigma_{ij}} = H \left( \frac{\partial m_{ij}}{\partial k} - \frac{1}{3} \frac{\partial m_{kl}}{\partial k} \delta_{kl} \delta_{ij} \right) \] (104.272)

\[ \frac{\partial \bar{\alpha}_{ij}}{\partial \alpha_{mn}} = H \left( \frac{\partial m_{ij}}{\partial \alpha_{mn}} - \frac{1}{3} \frac{\partial m_{kl}}{\partial \alpha_{mn}} \delta_{kl} \delta_{ij} \right) \] (104.273)

Armstrong-Frederick Kinematic Hardening for Drucker-Prager

\[ \bar{\alpha}_{ij} = \frac{2}{3} \frac{h_a}{p} m_{ij}^{\text{dev}} - c_r \left( \frac{2}{3} m_{rs} m_{rs} \right)^{0.5} \alpha_{ij} \] (104.274)

where \( p \) is pressure of the current stress state. The unit of parameter \( h_a \) is Pascal. The parameter \( c_r \) is unitless. The \( \alpha_{ij} \) in Drucker-Prager is unitless.

The pressure \( p \) is introduced for two reasons.

- The center of the yield surface is \( p \alpha_{ij} \), not \( \alpha_{ij} \). The kinematic hardening rule is to control the center of the yield surface.
The unit in equation (104.303) matches after the pressure $p$ is introduced.

When the derivative of backstress $\bar{\alpha}_{ij} = 0$, the tensor $\alpha_{ij}$ reaches the tensor limit.

$$\alpha_{ij}^{\text{lim}} = \sqrt{\frac{2}{3}} h_a \frac{m_{ij}^{\text{dev}}}{pc \sqrt{m_{rs}^{\text{dev}} m_{rs}^{\text{dev}}}}$$

(104.275)

Some useful tensor derivatives for Drucker-Prager $\bar{\alpha}$.

$$m_{ij}^{\text{dev}} = m_{ij} - \frac{1}{3} m_{kl} \delta_{kl} \delta_{ij}$$

(104.276)

- Useful tensor derivatives for Drucker-Prager $\bar{\alpha}$ with respect to $\sigma$.

$$\frac{\partial \bar{\alpha}_{ij}}{\partial \sigma_{mn}} = 2 \frac{h_a}{3} \frac{\partial m_{ij}^{\text{dev}}}{\partial \sigma_{mn}} + \frac{2}{9p^2} h_a m_{ij}^{\text{dev}} \delta_{mn} - \frac{2}{3} c_r m_{rs}^{\text{dev}} \frac{\partial m_{rs}^{\text{dev}}}{\partial \sigma_{mn}} \left( 2 m_{ij}^{\text{dev}} m_{kl}^{\text{dev}} \right)^{-0.5} \alpha_{ij}$$

(104.277)

where

$$\frac{\partial m_{ij}^{\text{dev}}}{\partial \sigma_{mn}} = \frac{\partial m_{ij}}{\partial \sigma_{mn}} - \frac{1}{3} \frac{\partial m_{ot}}{\partial \sigma_{mn}} \delta_{ot} \delta_{ij}$$

(104.278)

- Useful tensor derivatives for Drucker-Prager $\bar{\alpha}$ with respect to $\alpha$.

$$\frac{\partial \bar{\alpha}_{ij}}{\partial \alpha_{mn}} = 2 \frac{h_a}{3} \frac{\partial m_{ij}^{\text{dev}}}{\partial \alpha_{mn}} - \frac{2}{3} c_r m_{rs}^{\text{dev}} \frac{\partial m_{rs}^{\text{dev}}}{\partial \alpha_{mn}} \left( \frac{2}{3} m_{ij}^{\text{dev}} m_{kl}^{\text{dev}} \right)^{-0.5} \alpha_{ij} - c_r \left( \frac{2}{3} m_{pq}^{\text{dev}} m_{pq}^{\text{dev}} \right)^{0.5} \delta_{im} \delta_{jn}$$

(104.279)

where

$$\frac{\partial m_{ij}^{\text{dev}}}{\partial \alpha_{mn}} = \frac{\partial m_{ij}}{\partial \alpha_{mn}} - \frac{1}{3} \frac{\partial m_{ot}}{\partial \alpha_{mn}} \delta_{ot} \delta_{ij}$$

(104.280)

- Useful tensor derivatives for Drucker-Prager $\bar{\alpha}$ with respect to $k$.

$$\frac{\partial \bar{\alpha}_{ij}}{\partial k} = 2 \frac{h_a}{3} \frac{\partial m_{ij}^{\text{dev}}}{\partial k} - \sqrt{\frac{2}{3}} c_r \alpha_{ij} (m_{rs}^{\text{dev}} m_{rs}^{\text{dev}})^{-0.5} \frac{\partial m_{pq}^{\text{dev}}}{\partial k} m_{pq}^{\text{dev}}$$

(104.281)

where

$$\frac{\partial m_{ij}^{\text{dev}}}{\partial k} = \frac{\partial m_{ij}}{\partial k} - \frac{1}{3} \frac{\partial m_{ot}}{\partial k} \delta_{ot} \delta_{ij}$$

(104.282)

### 104.6.7.4 Federico’s Description of a Drucker–Prager Kinematic Hardenig Model

Presented is a concise description of nonlinear rotational kinematic hardening (Armstrong-Frederick) Drucker-Prager model (Lemaitre and Chaboche, 1990).

**Elastic behavior** The standard Hooke’s law has been assumed for the sake of simplicity.
Yield function  The yield locus is of the same kind described by Prevost (1985a) and Manzari and Dafalias (1997), i.e. conical and allowed to rotate around its apex (the centre of rotation coincides – for cohesionless materials – with the origin of the principal stress space):

\[ f = \sqrt{(s_{ij} - p\alpha_{ij}) (s_{ij} - p\alpha_{ij})} - \sqrt{\frac{2}{3}}kp = 0 \]  

(104.283)

in which \( p \) is the isotropic mean pressure, \( \alpha_{ij} \) the back–stress ratio tensor and \( k \) a constitutive surface parameter.\(^{62} \). While this latter governs the opening angle of the cone, \( \alpha_{ij} \) is a second–rank deviatoric tensor determining the yield locus rotation.

Plastic flow rule  As usual, the incremental plastic strain tensor can be expressed as:

\[ \varepsilon_{ij}^p = \dot{\lambda}m_{ij} \]  

(104.284)

where \( \dot{\lambda} \) is a scalar plastic multiplier and \( m_{ij} \) assigning the direction of the plastic flow. While in the associated version of the model \( m_{ij} \) would coincide with the stress gradient of the yield function (104.283), here the following non–associated flow rule has been adopted (Prevost, 1985a; Manzari and Dafalias, 1997):

\[ m_{ij} = \left( \frac{\partial f}{\partial \sigma_{ij}} \right)^{dev} - \frac{1}{3}D\delta_{ij} \]  

(104.285)

where the superscript \(^{dev} \) and \( \delta_{ij} \) denote the “deviatoric” tensor operator and the Kronecker hydrostatic tensor, respectively, and \( D \) is a dilatancy coefficient defined as:

\[ D = \xi \left( \sqrt{\frac{2}{3}}k_d - \sqrt{r_{mn}r_{mn}} \right) \]  

(104.286)

Here, we defined normalized deviatoric stress tensor \( r_{ij} = s_{ij}/p \).

Apparently, the flow rule (104.285) implies the deviatoric plastic strain increment to be associated, while non-associativeness holds for the volumetric component. The definition (104.286) requires two constitutive parameters to be identified, namely \( k_d \) and \( \xi \): the former represents the stress obliquity for the transition from contractive to dilative response; the latter quantitatively governs the volumetric plastic strain rate. Specifically, \( k_d \) denotes the existence of a “dilatancy surface”, the soil response being contractive for inner stress states and dilative otherwise; this surface – characterized by no volumetric plastic strain – is still a Drucker–Prager conical locus, fixed in the principal stress space and with an opening angle given by \( k_d \).

\(^{62} \)stresses are meant here to be effective
Armstrong–Frederick kinematic hardening rule The last ingredient in the model formulation is represented by the hardening rule for the internal variable $\alpha_{ij}$. Here, an Armstrong–Frederick hardening (Armstrong and Frederick, 1966; Lemaitre and Chaboche, 1990) has been introduced:

$$
\dot{\alpha}_{ij} = \frac{2}{3} h_a \left( \epsilon_{ij}^p \right)^{dev} - c_r \alpha_{ij} \sqrt{\frac{2}{3} \left( \epsilon_{rs}^p \right)^{dev} \left( \epsilon_{rs}^p \right)^{dev}}
$$

(104.287)

where $h_a$ and $c_r$ are two hardening constitutive parameters. Equation (104.287) yields a saturation–type evolution under deviatoric plastic straining, up to the achievement of a limit back–stress ratio $\alpha_{ij}^{lim}$. Starting from (104.287), it could be easily proven that:

$$
\left\| \alpha_{ij}^{lim} \right\| = \sqrt{\frac{2}{3} \frac{h_a}{c_r}}
$$

(104.288)

i.e. the norm of the limit tensor $\alpha_{ij}^{lim}$ exclusively depends on the $h_a/c_r$ ratio for any value of the Lode angle. The existence of an outer bound for $\alpha_{ij}$ implies all feasible stress states to lie within a so–called bounding surface, determined both by $\alpha_{ij}^{lim}$ and the opening of the yield locus $f = 0$ (104.283). The bounding surface, governing the shear strength of the material, is in this case a Lode–angle insensitive Drucker–Prager cone: in the simplest case of triaxial loading, the limit stress obliquity $M$ can be shown to equal:

$$
M = \left( \frac{q}{p} \right)_{\text{failure}} = k + \frac{h_a}{c_r}
$$

(104.289)

both in compression and extension. From the above analytical relationships it can be inferred that the material shear strength is ruled by the ratio $h_a/c_r$, whereas $c_r$ determines the evolution rate of the back–stress tensor $\alpha_{ij}$.

104.6.7.5 Han’s Description of Drucker–Prager Model with Armstrong–Frederick Kinematic Hardening

Presented is a new description of non-associated Drucker-Prager model with nonlinear Armstrong-Frederick kinematic hardening. Compared with the traditional description shown in earlier sections, the main difference in this description is taking into consideration of the symmetry nature of stress and strain tensors. As a result, some of the derivatives of plastic flow and internal variable are different. More importantly, this change leads to a consistent stiffness tensor that has minor symmetry.

Elastic Behavior The elastic behavior is modeled as linear elastic, following classic generalized Hook’s law. The elastic stiffness tensor is that for isotropic, linear elastic material (Equation 104.163).

$$
E_{ijkl} = \frac{E}{2(1+\nu)} \left( \frac{2\nu}{1-2\nu} \delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk} \right)
$$

(104.290)
**Yield Function**  The yield function and its derivatives remain unchanged.

\[
f = [(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})]^{0.5} - \sqrt{\frac{2}{3}} kp = 0 \tag{104.291}
\]

\[
\frac{\partial f}{\partial \sigma_{ij}} = \left[ (s_{ij} - p\alpha_{ij}) + \frac{1}{3} \alpha_{mn} \delta_{ij} (s_{mn} - p\alpha_{mn}) \right] [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} + \sqrt{\frac{2}{27}} k \delta_{ij} \tag{104.292}
\]

\[
\frac{\partial f}{\partial \alpha_{ij}} = -p (s_{ij} - p\alpha_{ij}) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} \tag{104.293}
\]

\[
\frac{\partial f}{\partial k} = -\sqrt{\frac{2}{3}} p \tag{104.294}
\]

**Plastic Flow**  The changes in derivatives of plastic flow, and later in hardening, comes from the clarification of derivative of the stress tensor with respect to itself. According to Park (2018), since the stress tensor is intrinsically symmetric, its derivative with respect to itself should be:

\[
\frac{\partial \sigma_{ij}}{\partial \sigma_{mn}} = \frac{1}{2} (\delta_{im}\delta_{jn} + \delta_{in}\delta_{jm}) \tag{104.295}
\]

Then, the derivative of deviatoric stress with respect to stress becomes:

\[
\frac{\partial s_{ij}}{\partial \sigma_{mn}} = \frac{1}{2} \delta_{im}\delta_{jn} + \frac{1}{2} \delta_{in}\delta_{jm} - \frac{1}{3} \delta_{mn}\delta_{ij} \tag{104.296}
\]

The non-associated plastic flow still has the same form:

\[
m_{ij} = \left( \frac{\partial f}{\partial \sigma_{ij}} \right)^{dev} = \frac{1}{3} D \delta_{ij} = (s_{ij} - p\alpha_{ij}) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5} - \frac{1}{3} D \delta_{ij} \tag{104.297}
\]

where

\[
D = \xi \left( \sqrt{\frac{2}{3}} k d - \sqrt{s_{mn}s_{mn}} \right) \tag{104.298}
\]

and

\[
\frac{\partial D}{\partial \sigma_{mn}} = -\xi \left[ p^{-1}s_{mn}(s_{kl}s_{kl})^{-0.5} + \frac{1}{3} p^{-2} \delta_{mn}(s_{ot}s_{ot})^{0.5} \right] \tag{104.299}
\]

Therefore,

\[
\frac{\partial m_{ij}}{\partial \sigma_{mn}} = \left( \frac{1}{2} \delta_{im}\delta_{jn} + \frac{1}{2} \delta_{in}\delta_{jm} - \frac{1}{3} \delta_{mn}\delta_{ij} + \frac{1}{3} \delta_{mn}\alpha_{ij} \right) [(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs})]^{-0.5}
\]

\[
- (s_{ij} - p\alpha_{ij}) \left[ (s_{mn} - p\alpha_{mn}) + \frac{1}{3} \alpha_{rs} (s_{rs} - p\alpha_{rs}) \delta_{mn} \right] [(s_{tu} - p\alpha_{tu})(s_{tu} - p\alpha_{tu})]^{-1.5}
\]

\[
+ \frac{1}{3} \xi \left[ s_{mn} p^{-1}(s_{kl}s_{kl})^{-0.5} + \frac{1}{3} \delta_{mn} p^{-2}(s_{ot}s_{ot})^{0.5} \right] \delta_{ij} \tag{104.300}
\]
\[
\frac{\partial m_{ij}}{\partial k} = 0 \tag{104.301}
\]

\[
\frac{\partial m_{ij}}{\partial \alpha_{mn}} = -p \left( \frac{1}{2} \delta_{mi} \delta_{nj} + \frac{1}{2} \delta_{mj} \delta_{ni} \right) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) \right]^{-0.5} + p (s_{ij} - p\alpha_{ij}) (s_{mn} - p\alpha_{mn}) \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) \right]^{-1.5} \tag{104.302}
\]

**Armstrong-Frederick kinematic hardening for Drucker-Prager**

\[
\bar{\alpha}_{ij} = \frac{2}{3} h_a \frac{m_{ij}^{dev}}{p_{atm}} - c_r \left( \frac{2}{3} m_{rs}^{dev} m_{rs}^{dev} \right)^{0.5} \alpha_{ij} \tag{104.303}
\]

where \( p_{atm} \) is the atmospheric pressure of 101.325 kPa. The unit of parameter \( h_a \) is Pascal. The parameter \( c_r \) is unitless. The \( \alpha_{ij} \) in Drucker-Prager is unitless. The atmospheric pressure \( p_{atm} \) is introduced so that the unit in equation (104.303) matches.

Notice that

\[
m_{rs}^{dev} m_{rs}^{dev} = (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) \right]^{-1} = 1 \tag{104.304}
\]

This means the hardening equation and related derivatives can be simplified:

\[
\bar{\alpha}_{ij} = \frac{2}{3} h_a \frac{m_{ij}^{dev}}{p_{atm}} - \sqrt{\frac{2}{3}} c_r \alpha_{ij} \tag{104.305}
\]

When the derivative of backstress \( \alpha_{ij} = 0 \), the tensor \( \alpha_{ij} \) reaches the tensor limit:

\[
\alpha_{ij}^\text{lim} = \sqrt{\frac{2}{3}} \frac{h_a}{p_{atm} c_r} m_{ij}^{dev} \tag{104.306}
\]

Some useful tensor derivatives for Drucker-Prager \( \bar{\alpha} \).

\[
m_{ij}^{dev} = (s_{ij} - p\alpha_{ij}) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) \right]^{-0.5} \tag{104.307}
\]

\[
\frac{\partial m_{ij}^{dev}}{\partial \sigma_{mn}} = \left( \frac{1}{2} \delta_{im} \delta_{jn} + \frac{1}{2} \delta_{jn} \delta_{im} - \frac{1}{3} \delta_{mn} \delta_{ij} + \frac{1}{3} \delta_{mn} \alpha_{ij} \right) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) \right]^{-0.5} - (s_{ij} - p\alpha_{ij}) \left[ (s_{mn} - p\alpha_{mn}) + \frac{1}{3} \alpha_{rs} (s_{rs} - p\alpha_{rs}) \delta_{mn} \right] \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) \right]^{-1.5} \tag{104.308}
\]

\[
\frac{\partial m_{ij}^{dev}}{\partial k} = 0 \tag{104.309}
\]

\[
\frac{\partial m_{ij}^{dev}}{\partial \alpha_{mn}} = -p \left( \frac{1}{2} \delta_{im} \delta_{jn} + \frac{1}{2} \delta_{jn} \delta_{im} \right) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) \right]^{-0.5} - p (s_{ij} - p\alpha_{ij}) (s_{mn} - p\alpha_{mn}) \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) \right]^{-1.5} \tag{104.310}
\]
• Tensor derivative of Drucker-Prager \( \bar{\alpha}_{ij} \) with respect to \( \sigma_{ij} \).
\[
\frac{\partial \bar{\alpha}_{ij}}{\partial \sigma_{mn}} = \frac{2h_a}{3p_{atm}} \frac{\partial m_{ij}^{dev}}{\partial \sigma_{mn}}
\] (104.311)

• Tensor derivative of Drucker-Prager \( \bar{\alpha}_{ij} \) with respect to \( k \).
\[
\frac{\partial \bar{\alpha}_{ij}}{\partial k} = 0
\] (104.312)

• Tensor derivative of Drucker-Prager \( \bar{\alpha}_{ij} \) with respect to \( \alpha_{ij} \).
\[
\frac{\partial \bar{\alpha}_{ij}}{\partial \alpha_{mn}} = \frac{2h_a}{3p_{atm}} \frac{\partial m_{ij}^{dev}}{\partial \alpha_{mn}} - \sqrt{\frac{2}{3}} c_r \left( \frac{1}{2} \delta_{im} \delta_{jn} + \frac{1}{2} \delta_{in} \delta_{jm} \right)
\] (104.313)

104.6.8 Hyperbolic Drucker Prager Model

104.6.8.1 Original Yield Function and Hyperbolic Function

The original Drucker-Prager yield function is
\[
\Phi(\sigma) = \sqrt{J_2} - \eta p - \xi
\] (104.314)

where
\[
\eta = \frac{3\tan \phi}{\sqrt{9 + 12\tan^2 \phi}}
\] (104.315)

\[
\xi = \frac{3c}{\sqrt{9 + 12\tan^2 \phi}}
\] (104.316)

where \( \phi \) is the friction ratio, and \( c \) is the cohesion. Besides, the constant \( \eta \) is unitless. The unit of constant \( \xi \) is Pascal. In addition, here we define that \( p \) is compression-positive. Namely, \( p = -\frac{\sigma_{ii}}{3} \).

Rewrite the yield function (104.314) with \( p, q \) such that
\[
\Phi(\sigma) = \frac{q}{\sqrt{3}} - \eta p - \xi
\] (104.317)

The slope of the yield surface is \( \sqrt{3} \eta \), and the hydrostatic cutoff point is at \( (-\frac{\xi}{\eta}, 0) \).

104.6.8.2 Hyperbolic Drucker Prager Model

Assume a standard hyperbolic function is
\[
\left( \frac{x - d}{a} \right)^2 - \left( \frac{y}{b} \right)^2 = 1
\] (104.318)

In the hyperbolic function, as showed in the Figure 104.9, the distance between the original apex to the rounded hydrostatic cut-off is \( a \), and the corresponding \( y \) value is \( b \). The hydrostatic cut off of the asymptotic line is \( d \).
104.6.8.3 Modified Hyperbolic Drucker Prager

**Mapping the Hyperbolic Equation.** In the modified hyperbolic Drucker Prager yield surface, the asymptotic line is the original yield surface.

The slope of the asymptotic line is

\[
\frac{b}{a} = \sqrt{3} \eta \tag{104.319}
\]

The hydrostatic cut-off of the asymptotic line is

\[
d = \frac{\xi}{\eta} \tag{104.320}
\]

The rounded distance \(a\) of the hyperbolic line is a new parameter.

The hyperbolic function is

\[
\left(\frac{p + \frac{\xi}{\eta}}{a}\right)^2 - \left(\frac{q}{\sqrt{3} a \eta}\right)^2 = 1 \tag{104.321}
\]

**Removing the Negative Branch.** To remove the negative branch, we need the pressure \((p + \frac{\xi}{\eta})\) always be positive. Therefore,

\[
\left(\frac{p + \frac{\xi}{\eta}}{a}\right)^2 = \left(\frac{q}{\sqrt{3} a \eta}\right)^2 + 1 \tag{104.322}
\]
Take the root on both sides,

\[
p + \frac{\xi}{\eta} = \sqrt{\left(\frac{q}{\sqrt{3} \ a \ \eta}\right)^2 + 1}
\]  

(104.323)

**Inside the hyperbolic** To make the yield surface value smaller than zero when the stress state is inside the modified cone, the yield surface is

\[
0 = \sqrt{\left(\frac{q}{\sqrt{3}} \ a \ \eta\right)^2 + 1} - \frac{p + \xi}{a}
\]  

(104.324)

**Avoiding Zero Denominator.** To avoid the zero denominator situation, multiply the equation on both sides by \(\eta\) and \(a\).

\[
0 = \sqrt{\left(\frac{q}{\sqrt{3}}\right)^2 + a^2\eta^2} - \eta p - \xi
\]  

(104.325)

**The basic yield function** The yield function now has one internal variable \(\eta\) for isotropic hardening.

\[
\Phi(\sigma, \eta) = \sqrt{\left(\frac{q}{\sqrt{3}}\right)^2 + a^2\eta^2} - \eta p - \xi
\]  

(104.326)

Simplify the yield function and substitute \(q = \sqrt{\frac{2}{3} s_{ij} s_{ij}}\), we have

\[
\Phi(\sigma, \eta) = \sqrt{\frac{1}{2} s_{ij} s_{ij} + a^2\eta^2} - \eta p - \xi
\]  

(104.327)

Introduce the capability of rotational kinematic hardening by \(\alpha\).

\[
\Phi(\sigma, \eta, \alpha) = \sqrt{\frac{1}{2} (s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij}) + a^2\eta^2} - \eta p - \xi
\]  

(104.328)

**104.6.8.4 The Non-Associative Plastic Potential Function.**

The non-associative plastic flow direction is

\[
m_{ij} = \left(\frac{\partial f}{\partial \sigma_{ij}}\right)^{dev} + \frac{1}{3} \eta \delta_{ij}
\]  

(104.329)

where \(\bar{\eta}\) controls the plastic flow direction. When \(\bar{\eta} = 0\), the material has the deviatoric plastic flow only.

The relation between the dilatancy angle and \(\bar{\eta}\) is similar to the relation between the friction angle and \(\eta\).

\[
\bar{\eta} = \frac{3\tan \psi}{\sqrt{9 + 12\tan^2 \psi}}
\]  

(104.330)

where \(\psi\) is the dilatancy angle.
Numerical Issues in the Non-Associative Plastic Potential Function. The non-associative plastic flow rule will have asymmetric stiffness matrix, which requires a asymmetric solver. The non-associative requires smaller subincrements for convergence. One important non-associative plastic potential is the purely deviatoric plastic flow.

The purely deviatoric plastic flow may have the numerical issues for the convergence. For example, in Fig 104.10, the purely deviatoric plastic flow direction can never return back to the yield surface. Subincrements can solve this problem.

Figure 104.10: The inconvergence situation for purely deviatoric plastic flow.

104.6.8.5 Han’s Description of Hyperbolic Drucker–Prager Model with Armstrong–Frederick Kinematic Hardening

Presented is a new description of non-associated hyperbolic Drucker-Prager model with nonlinear Armstrong-Frederick kinematic hardening. It’s intended to keep the same theoretical framework and the same set of material parameters as the Drucker-Prager material model shown in section 104.6.7.5.

Classic Drucker–Prager Yield Function with Cohesion This is the yield function shown in Equation 104.237, extended to considering cohesion

\[ f = q - Mp - \sqrt{3}\beta \]  

(104.331)

where

\[ M = \frac{6 \sin \phi}{3 - \sin \phi} \quad \text{and} \quad \beta = \frac{6 \cos \phi_0}{\sqrt{3}(3 - \sin \phi_0)} c \]  

(104.332)
Note that $\phi$ is the friction angle, which can evolve if hardening is present in the model, and $\phi_0$ is the initial friction angle.

Rewrite Equation 104.333 using the parameters in Real-ESSI implementation

$$f = \sqrt{(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})} - \sqrt{\frac{2}{3}kp - \sqrt{2}\beta} \quad (104.333)$$

**Hyperbolic Drucker–Prager Yield Function** Assume the same generic hyperbolic function shown in Equation 104.318, the hyperbolic Drucker-Prager yield function considering isotropic and kinematic hardening is given as

$$f = \sqrt{(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})} + \frac{2}{3}k^2a^2 - \sqrt{\frac{2}{3}kp - \sqrt{2}\beta} \quad (104.334)$$

where $a$ is the rounded distance shown in Figure 104.9.

Derivatives of the yield function Equation 104.334 are shown below.

$$\frac{\partial f}{\partial \sigma_{ij}} = \left[ (s_{ij} - p\alpha_{ij}) + \frac{1}{3}\alpha_{kl}(s_{kl} - p\alpha_{kl})\delta_{ij} \right] \left[ (s_{mn} - p\alpha_{mn}) (s_{mn} - p\alpha_{mn}) + \frac{2}{3}k^2a^2 \right]^{-0.5} - \frac{1}{3}D\delta_{ij} \quad (104.335)$$

$$\frac{\partial f}{\partial \alpha_{ij}} = -p(s_{ij} - p\alpha_{ij}) \left[ (s_{mn} - p\alpha_{mn}) (s_{mn} - p\alpha_{mn}) + \frac{2}{3}k^2a^2 \right]^{-0.5} \quad (104.336)$$

$$\frac{\partial f}{\partial k} = \frac{2}{3}ka^2 \left[ (s_{mn} - p\alpha_{mn}) (s_{mn} - p\alpha_{mn}) + \frac{2}{3}k^2a^2 \right]^{-0.5} - \sqrt{\frac{2}{3}p} \quad (104.337)$$

**Plastic Flow** The non-associated plastic flow is defined as

$$m_{ij} = \left( \frac{\partial f}{\partial \sigma_{ij}} \right)^{dev} - \frac{1}{3}D\delta_{ij} = (s_{ij} - p\alpha_{ij}) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) + \frac{2}{3}k^2a^2 \right]^{-0.5} - \frac{1}{3}D\delta_{ij} \quad (104.338)$$

where

$$D = \xi \left( \sqrt{\frac{2}{3}kp - \sqrt{s_{mn}s_{mn}}/p} \right) \quad (104.339)$$

The derivatives of the plastic flow are shown below.

$$\frac{\partial m_{ij}}{\partial \sigma_{mn}} = \left( \frac{1}{2}\delta_{im}\delta_{jn} + \frac{1}{2}\delta_{in}\delta_{jm} - \frac{1}{3}\delta_{mn}\delta_{ij} + \frac{1}{3}\delta_{mn}\alpha_{ij} \right) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) + \frac{2}{3}k^2a^2 \right]^{-0.5} - (s_{ij} - p\alpha_{ij}) \left[ (s_{mn} - p\alpha_{mn}) + \frac{1}{3}\alpha_{rs}(s_{rs} - p\alpha_{rs})\delta_{mn} \right] \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) + \frac{2}{3}k^2a^2 \right]^{-1.5} \quad (104.340)$$
\[ \frac{\partial m_{ij}}{\partial \alpha_{mn}} = -p \left( \frac{1}{2} \delta_{mi} \delta_{nj} + \frac{1}{2} \delta_{mj} \delta_{ni} \right) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) + \frac{2}{3} k^2 a^2 \right]^{-0.5} \]

\[ + p (s_{ij} - p\alpha_{ij}) (s_{mn} - p\alpha_{mn}) \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) + \frac{2}{3} k^2 a^2 \right]^{-1.5} \]  

(104.341)

\[ \frac{\partial m_{ij}}{\partial k} = -\frac{2}{3} k a^2 (s_{ij} - p\alpha_{ij}) (s_{mn} - p\alpha_{mn}) \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) + \frac{2}{3} k^2 a^2 \right]^{-1.5} \]  

(104.342)

**Linear Isotropic Hardening**  Linear isotropic hardening is used for this material model. The evolution of the internal variable \( k \) is defined as

\[ \bar{k} = H m^{equi}, \quad m^{equi} = \left( \frac{2}{3} m_{ij} m_{ij} \right)^{0.5} \]  

(104.343)

where \( H \) is a material constant.

The derivatives of the internal variable \( k \) are shown below.

\[ \frac{\partial \bar{k}}{\partial \sigma_{ij}} = \frac{2H}{3} m_{kl} \frac{\partial m_{kl}}{\partial \sigma_{ij}} \left( m^{equi} \right)^{-1} \]  

(104.344)

\[ \frac{\partial \bar{k}}{\partial \alpha_{ij}} = \frac{2H}{3} m_{kl} \frac{\partial m_{kl}}{\partial \alpha_{ij}} \left( m^{equi} \right)^{-1} \]  

(104.345)

\[ \frac{\partial \bar{k}}{\partial k} = \frac{2H}{3} m_{kl} \frac{\partial m_{kl}}{\partial k} \left( m^{equi} \right)^{-1} \]  

(104.346)

**Armstrong-Frederick Kinematic Hardening for Hyperbolic Drucker-Prager**

\[ \bar{\alpha}_{ij} = \frac{2}{3} \frac{h_a}{p_{atm}} m^{dev}_{ij} - c_r \left( \frac{2}{3} m^{dev}_{rs} m^{dev}_{rs} \right)^{0.5} \bar{\alpha}_{ij} \]  

(104.347)

where \( p_{atm} \) is the atmospheric pressure of 101.325 kPa. The unit of parameter \( h_a \) is Pascal. The parameter \( c_r \) is unitless. The \( \bar{\alpha}_{ij} \) in Drucker-Prager is unitless.

Some useful tensor derivatives for Drucker-Prager \( \bar{\alpha} \).

\[ m^{dev}_{ij} = (s_{ij} - p\alpha_{ij}) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) + \frac{2}{3} k^2 a^2 \right]^{-0.5} \]  

(104.348)

\[ \frac{\partial m^{dev}_{ij}}{\partial \sigma_{mn}} = \left( \frac{1}{2} \delta_{im} \delta_{jn} + \frac{1}{2} \delta_{jm} \delta_{ni} \right. \left. - \frac{1}{3} \delta_{mn} \delta_{ij} + \frac{1}{3} \delta_{mn} \alpha_{ij} \right) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) + \frac{2}{3} k^2 a^2 \right]^{-0.5} \]

\[ - (s_{ij} - p\alpha_{ij}) \left[ (s_{mn} - p\alpha_{mn}) + \frac{1}{3} \alpha_{rs} (s_{rs} - p\alpha_{rs}) \delta_{mn} \right] \left( s_{tu} - p\alpha_{tu} \right) \left( s_{tu} - p\alpha_{tu} \right) + \left. \frac{2}{3} k^2 a^2 \right]^{-1.5} \]  

(104.349)
\[
\frac{\partial m_{ij}^{\text{dev}}}{\partial \sigma_{mn}} = -p \left( \frac{1}{2} \delta_{im} \delta_{jn} + \frac{1}{2} \delta_{in} \delta_{jm} \right) \left[ (s_{rs} - p\alpha_{rs}) (s_{rs} - p\alpha_{rs}) + \frac{2}{3} k^2 a^2 \right]^{-0.5} \\
\quad - p (s_{ij} - p\alpha_{ij}) (s_{mn} - p\alpha_{mn}) \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) + \frac{2}{3} k^2 a^2 \right]^{-1.5}
\]

(104.350)

\[
\frac{\partial m_{ij}^{\text{dev}}}{\partial k} = -\frac{2}{3} k a^2 (s_{ij} - p\alpha_{ij}) (s_{mn} - p\alpha_{mn}) \left[ (s_{tu} - p\alpha_{tu}) (s_{tu} - p\alpha_{tu}) + \frac{2}{3} k^2 a^2 \right]^{-1.5}
\]

(104.351)

Tensor derivative of Drucker-Prager \(\bar{\alpha}_{ij}\) with respect to \(\sigma_{ij}\).

\[
\frac{\partial \bar{\alpha}_{ij}}{\partial \sigma_{mn}} = \frac{2 h_a}{3 p_{atm}} \frac{\partial m_{ij}^{\text{dev}}}{\partial \sigma_{mn}} - \frac{2}{3} c_r m_{kl} \frac{\partial m_{kl}^{\text{dev}}}{\partial \sigma_{mn}} \left( \frac{2}{3} m_{rs}^\text{dev} m_{rs}^\text{dev} \right)^{-0.5} \alpha_{ij}
\]

(104.352)

Tensor derivative of Drucker-Prager \(\bar{\alpha}_{ij}\) with respect to \(\alpha_{ij}\).

\[
\frac{\partial \bar{\alpha}_{ij}}{\partial \alpha_{mn}} = \frac{2 h_a}{3 p_{atm}} \frac{\partial m_{ij}^{\text{dev}}}{\partial \alpha_{mn}} - \frac{2}{3} c_r m_{kl} \frac{\partial m_{kl}^{\text{dev}}}{\partial \alpha_{mn}} \left( \frac{2}{3} m_{rs}^\text{dev} m_{rs}^\text{dev} \right)^{-0.5} \alpha_{ij}
\]

\[
- c_r \left( \frac{2}{3} m_{rs}^\text{dev} m_{rs}^\text{dev} \right)^{0.5} \left( \frac{1}{2} \delta_{mi} \delta_{nj} + \frac{1}{2} \delta_{nj} \delta_{mi} \right)
\]

(104.353)

Tensor derivative of Drucker-Prager \(\bar{\alpha}_{ij}\) with respect to \(k\).

\[
\frac{\partial \bar{\alpha}_{ij}}{\partial k} = \frac{2 h_a}{3 p_{atm}} \frac{\partial m_{ij}^{\text{dev}}}{\partial k} - \frac{2}{3} c_r m_{kl} \frac{\partial m_{kl}^{\text{dev}}}{\partial k} \left( \frac{2}{3} m_{rs}^\text{dev} m_{rs}^\text{dev} \right)^{-0.5} \alpha_{ij}
\]

(104.354)

**104.6.9 Rounded Mohr-Coulomb Model**

**DSL COMMANDS** for the rounded Mohr-Coulomb material models are given in section 205.3 on page 791.

The model consists of a smooth conical yield surface. This conical yield surface has an apex that is located at the zero of stress coordinate system or to the extension side, depending on the cohesion characteristic of material in question. The yield surface is defined by:

\[
F(p, q, \theta, \kappa_{\text{cone}}) = q \left( 1 + \frac{q}{q_a} \right)^m \left[ g(\theta) - \eta_{\text{cone}} (\kappa_{\text{cone}}) (p - p_c) \right]
\]

(104.355)

where \(q_a\) is a (positive) reference deviator stress, \(m\) is a material constant such that \(0 \leq m \leq 1\), controlling the curvature of the cone in the meridian \((p, q)\) planes, \(\eta_{\text{cone}}\) represents the angle of internal friction, and \(p_c\) represents the cohesion. \(\eta_{\text{cone}}\) is a function of the hardening variable \(\kappa_{\text{cone}}\) which, in turn, is a function of the plastic work. Moreover, an asymmetric trace of the yield surface in the deviatoric plane is generated through the introduction of \(g(\theta)\) according to the expression by Willam and Warnke (1974):
\[ g(\theta) = \frac{4(1-e^2) \cos^2 \theta + (2e-1)^2}{2(1-e^2) \cos \theta + (2e-1) \sqrt{4(1-e^2) \cos^2 \theta + 5e^2 - 4e}} \] (104.356)

and \( e \) is an eccentricity parameter that satisfies the condition \( 1/2 < e \leq 1 \). From eqn. (104.356) we conclude that \( g(0) = 1/e \) and \( g(\pi/3) = 1 \). For \( e = 1 \Rightarrow g(\theta) = 1 \), so the influence of the third stress invariant via \( \theta \) is dropped and the conical surface represents a curved cone\(^{63}\). For \( e \to 1/2 \) the triangular cone\(^{64}\) is obtained. A nice pictorial representation of function \( 1/g(\theta) \) is presented in the Figure (104.6.9), showing material model traces in the deviatoric plane.

![Figure 104.11: Willam Warnke function 1/g for different values of e (e = 0.51, 0.6, 0.7, 0.8, 0.9, 1.0 ) representing traces of rounded Mohr-Coulomb model in deviatoric space.](image)

### 104.6.10 Modified Cam-Clay Model

**DSL COMMANDS** for the Modified Cam Clay material models are given in section 205.3 on page 791.

The pioneering research work on the critical state soil mechanics by the researchers at Cambridge University (Roscoe et al., 1958), (Roscoe and Burland, 1968), (Muir Wood, 1990)) has made great contribution on the modern soil elastoplastic models. The original Cam Clay model (Roscoe et al.,

\(^{63}\) extended Drucker Prager cone.
\(^{64}\) extended Rankine yield criteria.
1963), and later the modified Cam Clay model (Schofield and Wroth, 1968) were within the critical state soil mechanics framework. We focus on only the modified Cam Clay model and herein the word ‘modified’ is omitted to shorten writing.

### 104.6.10.1 Critical State

The critical state line (CSL) takes the form

\[
e_c = e_{c,r} - \lambda_c \ln p_c
\]  

(104.357)

where \(e_c\) is the critical void ratio at the critical mean effective stress \(p_c\), \(e_{c,r}\) is the reference critical void ratio, \(\lambda_c\) is the normal consolidation slope.

The critical state soil mechanics assumes that the normal consolidation line (NRL) is parallel to the CSL, which is expressed by

\[
e = e_\lambda - \lambda \ln p
\]  

(104.358)

where \(e_\lambda\) is the intercept on the NRL at \(p = 1\). \(\lambda\) is the normal consolidation slope or the elastoplastic slope of \(e - \ln p\) relation, and \(\lambda_c = \lambda\).

The unloading-reloading line (URL) take the similar form but with different slope by

\[
e = e_\kappa - \kappa \ln p
\]  

(104.359)

where \(e_\kappa\) is the intercept on the URL at \(p = 1\). \(\lambda_c\) is the normal consolidation slope or the elastoplastic slope of \(e - \ln p\) relation.

### 104.6.10.2 Elasticity

The elastic bulk modulus \(K\) can be directly derived from the Equation 104.359 and takes the form

\[
K = \frac{(1 + e)p}{\kappa}
\]  

(104.360)

If a constant Poisson’s ratio \(\nu\) is assumed, since the isotropic elasticity needs only two material constants, the shear elastic modulus can be obtained in terms of \(K\) and \(\nu\) by

\[
G = \frac{3(1 - 2\nu)(1 + e)}{2(1 + \nu)\kappa} p
\]  

(104.361)

Alternatively, a constant shear elastic modulus \(G\) can be assumed and then the Poisson’s ratio \(\nu\) is expressed in terms of \(K\) and \(G\) as

\[
\nu = \frac{3K - 2G}{2(G + 3K)}
\]  

(104.362)

\(^{65}\)In this chapter, only single-phase (dry phase) is studied, the total and effective stresses are thus identical, e.g. \(p'_c = p_c\).
104.6.10.3 Yield Function

The yield function of the Cam Clay model is defined by

\[ f = q^2 - M_c^2 [p(p_0 - p)] = 0 \]  \hspace{1cm} (104.363)

where \( M_c \) is the critical state stress ratio in the \( q - p \) plane, and the \( p_0 \) is the initial internal scalar variable, which is controlled by the change of the plastic volumetric strain.

The gradient of the yield surface to the stress can be obtained as

\[ \frac{\partial f}{\partial \sigma_{ij}} = 2q \frac{\partial q}{\partial \sigma_{ij}} - M_c^2 (2p - p_0) \frac{\partial p}{\partial \sigma_{ij}} = 3s_{ij} + \frac{1}{3} M_c^2 (p_0 - 2p) \delta_{ij} \]  \hspace{1cm} (104.364)

where \( \partial q/\partial \sigma_{ij} \) and \( \partial p/\partial \sigma_{ij} \) are independent of the yield function.

The gradient of the yield surface to \( p_0 \) will be used in the integration algorithm, and can be expressed by

\[ \frac{\partial f}{\partial p_0} = M_c^2 p \]  \hspace{1cm} (104.365)

104.6.10.4 Plastic Flow

The plastic flow of the Cam Clay model is associated with its yield function, in other words, the plastic flow is defined by the potential function, \( g \), which is assumed the same as the yield function, \( f \).

\[ g = f = q^2 - M_c^2 [p(p_0 - p)] = 0 \]  \hspace{1cm} (104.366)

The stress gradient to the yield surface can be obtained as

\[ m_{ij} = \frac{\partial g}{\partial \sigma_{ij}} = 2q \frac{\partial q}{\partial \sigma_{ij}} + M_c^2 (2p - p_0) \frac{\partial p}{\partial \sigma_{ij}} = 3s_{ij} + \frac{1}{3} M_c^2 (p_0 - 2p) \delta_{ij} \]  \hspace{1cm} (104.367)

It can define the plastic dilation angle \( \beta \), which is related to the ratio of plastic volumetric and deviatoric strain (Muir Wood, 1990), by

\[ \tan \beta = -\frac{\Delta \varepsilon_v^p}{\Delta \varepsilon_q^p} = \frac{M_c^2 (p_0 - 2p)}{2q} \]  \hspace{1cm} (104.368)

It is interesting to find that from Equation 104.368, when \( p < p_0/2 \), the plastic dilation angle is positive; when \( p > p_0/2 \), the plastic dilation angle is negative. If \( p = p_0/2 \), the plastic dilation angle is zero, which is corresponding to the critical state. This is evidently more realistic than Drucker-Prager model, whose associated plastic flow always gives positive plastic dilation angle.
104.6.10.5 Evolution Law

The evolution law of the Cam Clay model is a scalar one, which can be expressed by

\[
\dot{p}_0 = \frac{(1 + e)p_0}{\lambda - \kappa} \dot{e}_v^p
\]

(104.369)

With this scalar evolution law, the change of \( p_0 \) is decided by the change of plastic volumetric strain. When it reaches the critical state, or when there is no plastic volumetric strain, the evolution of \( p_0 \) will cease. From Equation 104.369, one gets

\[
\dot{p}_0 = \frac{(1 + e)p_0}{\lambda - \kappa} m_{ii}
\]

(104.370)

so if using Equation 104.367 further, one obtains

\[
h = \frac{(1 + e)p_0}{\lambda - \kappa} M_c^2 (2p - p_0)
\]

(104.371)

or by dilation angle,

\[
h = \frac{2(1 + e)p_0 q}{\lambda - \kappa} \tan \beta
\]

(104.372)

104.6.10.6 Yield and Plastic Potential Functions: Cam-Clay Model

Yield function and related derivatives

\[
f = q^2 - M_c^2 [p(p_0 - p)] = 0
\]

(104.373)

\[
\frac{\partial f}{\partial \sigma_{ij}} = 2q \frac{\partial q}{\partial \sigma_{ij}} - M_c^2 (2p - p_0) \frac{\partial p}{\partial \sigma_{ij}} = 3s_{ij} + \frac{1}{3} M_c^2 (p_0 - 2p) \delta_{ij}
\]

(104.374)

\[
\frac{\partial f}{\partial p_0} = -M_c^2 p
\]

(104.375)

Plastic flow (associated plasticity) and related derivatives

\[
m_{ij} = \frac{\partial f}{\partial \sigma_{ij}} = 3s_{ij} + \frac{1}{3} M_c^2 (p_0 - 2p) \delta_{ij}
\]

(104.376)

\[
\frac{\partial m_{ij}}{\partial \sigma_{mn}} = 3 \frac{\partial s_{ij}}{\partial \sigma_{mn}} - \frac{2}{3} M_c^2 \delta_{ij} \frac{\partial p}{\partial \sigma_{mn}} = 3 \delta_{im} \delta_{jn} - \delta_{ij} \delta_{mn} + \frac{2}{9} M_c^2 \delta_{ij} \delta_{mn}
\]

(104.377)
\[
\frac{\partial m_{ij}}{\partial p_0} = \frac{1}{3} M_c^2 \delta_{ij}
\]  

(104.378)

**Isotropic Hardening and related derivatives (CC_Ev)**  
Note the due to the current definition of \( p \) (i.e. \( p = -\frac{1}{3} \sigma_{ii} \)), a minus sign appears in from of the evolution of \( p_0 \) as follows:

\[
\bar{p}_0 = -\left(1 + e\right) p_0 \frac{m_{ii}}{\lambda - \kappa}
\]

(104.379)

\[
\bar{p}_0 = \frac{(1 + e) p_0}{\lambda - \kappa} M_c^2 (2p - p_0)
\]

(104.380)

\[
\frac{\partial \bar{p}_0}{\partial \sigma_{ij}} = \frac{(1 + e) p_0}{\lambda - \kappa} M_c^2 \left( -\frac{2}{3} \delta_{ij} \right)
\]

(104.381)

\[
\frac{\partial \bar{p}_0}{\partial p_0} = \frac{2 (1 + e)}{\lambda - \kappa} M_c^2 (p - p_0)
\]

(104.382)

### 104.6.11 SaniSand2004 (aka Dafalias-Manzari) Model

**DSL COMMANDS** for the Dafalias-Manzari material models are given in section 205.3 on page 791.

Within the critical state soil mechanics framework, Manzari and Dafalias (1997) proposed a two-surface sand model. This model considered the effects of the state parameter on the behaviors of the dense or loose sands. The features of this model include successfully predicting the softening at the dense state in drained loading, and also softening at the loose state but in the undrained loading. Dafalias and Manzari (2004a) later presented an improved version. This version introduced the fabric dilatancy tensor which has a significant effect on the contraction unloading response. It is also considered the Lode's angle effect on the bounding surface, which produces more realistic responses in non-triaxial conditions. Here only the new version is summarized. The compression stress is assumed negative here, which is different from the original reference by Dafalias and Manzari (2004a).

#### 104.6.11.1 Critical State

Instead of using the most common linear line of critical void ration vs. logarithmic critical mean effective stress, the power relation recently suggested by Li and Wang (1998) was used:

\[
e_c = e_{c,r} - \lambda_c \left( \frac{p_c}{P_{at}} \right)^\xi
\]

(104.383)

where \( e_c \) is the critical void ratio at the critical man effective stress \( p'_c \), \( e_{c,r} \) is the reference critical void ratio, \( \lambda_c \) and \( \xi \) (for most sands, \( \xi = 0.7 \)) are material constants, and \( P_{at} \) is the atmospheric pressure for normalization.
104.6.11.2 Elasticity

The elastic incremental moduli of shear and bulk, are following Richart et al. (1970):

\[ G = G_0 \frac{(2.97 - e)^2}{(1 + e)} \left( \frac{p}{P_{at}} \right)^{0.5} P_{at}, \quad K = \frac{2(1 + \nu)}{3(1 - 2\nu)} G \]  

where \( G_0 \) is a material constant, \( e \) is the void ratio, and \( \nu \) is the Poisson's ratio.

The isotropic hypoelasticity is then defined by

\[ \dot{\varepsilon}_{ij}^e = \frac{\dot{s}_{ij}}{2G}, \quad \dot{\varepsilon}_{v}^e = \frac{\dot{p}}{K} \]  

104.6.11.3 Yield Function

The yield function is defined by

\[ f = |\Lambda| - \sqrt{\frac{2}{3}} m_p = 0 \]  

where \( s_{ij} \) is the deviatoric stress tensor, \( \alpha_{ij} \) is the deviatoric back stress-ratio tensor, \( m \) is a material constant, and

\[ |\Lambda| = \|s_{ij} - p\alpha_{ij}\| = \left[ (s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij}) \right]^{0.5} \]  

The gradient of the yield surface to the stress can be obtained as

\[ \frac{\partial f}{\partial \sigma_{ij}} = n_{ij} + \frac{1}{3}(\alpha_{pq}n_{pq} + \sqrt{\frac{2}{3}} m)\delta_{ij} \]  

where \( r_{ij} = s_{ij}/p \) is the normalized deviatoric stress tensor, and \( n_{ij} \) is the unit gradient tensor to the yield surface defined by

\[ n_{ij} = \frac{s_{ij} - p\alpha_{ij}}{|\Lambda|} \]  

It is evident that \( n_{ii} \equiv 0 \) and \( n_{ij}n_{ij} \equiv 1 \).

The gradient of the yield surface to \( \alpha_{ij} \) can be easily obtained as

\[ \frac{\partial f}{\partial \alpha_{ij}} = -mn_{ij} \]  

The tensor of \( n_{ij} \) is to defined \( \theta_n \), the Lode’s angle of the yield gradient, by the equation

\[ \cos 3\theta_n = -\sqrt{6}n_{ij}n_{jk}n_{ki} \]  

where \( 0 \leq \theta_n \leq \pi/6 \) and \( \theta_n = 0 \) at triaxial compression and \( \theta_n = \pi/6 \) at triaxial extension.
The critical stress ratio $M$ at any stress state can be interpolated between $M_c$, the triaxial compression critical stress ratio, and $M_e$, the triaxial extension critical stress ratio.

$$M = M_c g(\theta_n, c), \quad g(\theta_n, c) = \frac{2c}{(1 + c) - (1 - c) \cos 3\theta_n}, \quad c = \frac{M_e}{M_c}$$  \hspace{1cm} (104.392)

The line from the origin of the $\pi$ plane parallel to $n_{ij}$ will intersect the bounding, critical and dilation surfaces at three ‘image’ back-stress ratio tensor $\alpha_b^{ij}$, $\alpha_c^{ij}$, and $\alpha_d^{ij}$ respectively (Figure 104.12), which are expressed as

$$\alpha_b^{ij} = \sqrt{\frac{2}{3}} [M \exp(-n^b \psi) - m] n_{ij} = \left( \sqrt{\frac{2}{3}} \alpha_b^\theta \right) n_{ij}$$  \hspace{1cm} (104.393)

$$\alpha_c^{ij} = \sqrt{\frac{2}{3}} [M - m] n_{ij} = \left( \sqrt{\frac{2}{3}} \alpha_c^\theta \right) n_{ij}$$  \hspace{1cm} (104.394)

$$\alpha_d^{ij} = \sqrt{\frac{2}{3}} [M \exp(n^d \psi) - m] n_{ij} = \left( \sqrt{\frac{2}{3}} \alpha_d^\theta \right) n_{ij}$$  \hspace{1cm} (104.395)

where $\psi = e - e_c$ is the state parameter; $n^b$ and $n^d$ are material constants.

Figure 104.12: Schematic illustration of the yield, critical, dilatancy, and bounding surfaces in the $\pi$-plane of deviatoric stress ratio space (after Dafalias and Manzari 2004).
104.6.11.4 Plastic Flow

The plastic strain is given by
\[ \dot{\varepsilon}_{ij}^p = \dot{\lambda}R_{ij} = \dot{\lambda}(R'_{ij} + \frac{1}{3}D\delta_{ij}) \] (104.396)

The deviatoric plastic flow tensor is
\[ R'_{ij} = Bn_{ij} + C(n_{ik}n_{kj} - \frac{1}{3}\delta_{ij}) \] (104.397)
where
\[ B = 1 + \frac{3}{2} \frac{1-c}{c} g \cos \theta_n, \quad C = 3\sqrt{\frac{3}{2}} \frac{1-c}{c} g \] (104.398)

The volumetric plastic flow part is
\[ D = -A_d(\alpha_{ij}^d - \alpha_{ij})n_{ij} = -A_d\left(\sqrt{\frac{2}{3}} \alpha_{ij}^d - \alpha_{ij}n_{ij}\right) \] (104.399)
where
\[ A_d = A_0(1 + \langle z_{ij}n_{ij}\rangle) \] (104.400)

\( A_0 \) is a material constant, and \( z_{ij} \) is the fabric dilation tensor. The Macauley brackets \( \langle \rangle \) is defined that
\[ \langle x \rangle = x, \text{ if } x > 0 \quad \text{and} \quad \langle x \rangle = 0, \text{ if } x \leq 0. \]

104.6.11.5 Evolution Laws

This model has two tensorial evolution internal variable, namely, the back stress-ratio tensor \( \alpha_{ij} \) and the fabric dilation tensor \( z_{ij} \).

The evolution law for the back stress-ratio tensor \( \alpha_{ij} \) is
\[ \dot{\alpha}_{ij} = \dot{\lambda}\left[\frac{2}{3}h(\alpha_{ij}^b - \alpha_{ij})\right] \] (104.401)
with
\[ h = \frac{b_0}{(\alpha_{ij} - \bar{\alpha}_{ij})n_{ij}} \] (104.402)

where \( \bar{\alpha}_{ij} \) is the initial value of \( \alpha_{ij} \) at initiation of a new loading process and is updated to the new value when the denominator of Equation 104.402 becomes negative. \( b_0 \) is expressed by
\[ b_0 = G_0h_0(1 - c_he)\left(\frac{p}{P_{at}}\right)^{-0.5} \] (104.403)

where \( h_0 \) and \( c_h \) are material constants.

The evolution law for the fabric dilation tensor \( z_{ij} \) is
\[ \dot{z}_{ij} = -c_z\langle \dot{D} \rangle(z_{\text{max}}n_{ij} + z_{ij}) \] (104.404)
where \( c_z \) and \( z_{\text{max}} \) are material constants.
104.6.11.6 Analytical Derivatives for the Implicit Algorithm

When implemented into an implicit algorithm for the Dafalias-Manzari model, some complicated additional analytical derivatives are needed. This section gives the analytical derivatives expressions based on the tensor calculus.

Analytical expression of $\frac{\partial m_{ij}}{\partial \sigma_{kl}}$:

\[
\frac{\partial m_{ij}}{\partial \sigma_{mn}} = B \frac{\partial n_{ij}}{\partial \sigma_{mn}} + n_{ij} \frac{\partial B}{\partial \sigma_{mn}} + C \frac{\partial n_{ik}}{\partial \sigma_{mn}} n_{kj} + (n_{ik} n_{kj} - \frac{1}{3} \delta_{ij}) \frac{\partial C}{\partial \sigma_{mn}} + \frac{1}{3} \delta_{ij} \frac{\partial D}{\partial \sigma_{mn}}
\]

(104.405)

where

\[
\frac{\partial n_{ij}}{\partial \sigma_{mn}} = \frac{1}{|\Lambda|} \left[ I_{ijmn} - \frac{1}{3} \delta_{ij} \delta_{mn} + \frac{1}{3} \alpha_{ij} \delta_{mn} - n_{ij} n_{mn} - \frac{1}{3} (\alpha_{ab} n_{ab}) n_{ij} \delta_{mn} \right]
\]

(104.406)

\[
\frac{\partial D}{\partial \sigma_{mn}} = - \frac{\partial A_d}{\partial \sigma_{mn}} \left( \sqrt{\frac{2}{3}} \alpha_d^{\theta} - \alpha_{ab}^{\theta} n_{ab} \right) - A_d \left( \sqrt{\frac{2}{3}} \frac{\partial \alpha_d^{\theta}}{\partial \sigma_{mn}} - \alpha_{ab}^{\theta} \frac{\partial n_{ab}}{\partial \sigma_{mn}} \right)
\]

(104.407)

and

\[
\frac{\partial B}{\partial \sigma_{mn}} = \frac{3}{2} \left( \frac{1 - c}{c} \right) \left( \frac{\partial g}{\partial \sigma_{mn}} \cos 3\theta + g \frac{\partial \cos 3\theta}{\partial \sigma_{mn}} \right)
\]

(104.408)

\[
\frac{\partial C}{\partial \sigma_{mn}} = 3 \sqrt{\frac{3}{2}} \left( \frac{1 - c}{c} \right) \frac{\partial \alpha_d^{\theta}}{\partial \sigma_{mn}}
\]

(104.409)

\[
\frac{\partial \alpha_d^{\theta}}{\partial \sigma_{mn}} = M_c \exp \left( n^d \psi \right) \left( gn^d \frac{\partial \psi}{\partial \sigma_{mn}} + \frac{\partial g}{\partial \sigma_{mn}} \right)
\]

(104.410)

\[
\frac{\partial \psi}{\partial \sigma_{mn}} = - \frac{\xi \lambda_c}{3 p_{at}} \left( \frac{p}{p_{at}} \right)^{(\xi-1)} \delta_{mn}
\]

(104.411)

\[
\frac{\partial g}{\partial \sigma_{mn}} = g^2 \left( \frac{1 - c}{2c} \right) \frac{\partial \cos 3\theta}{\partial \sigma_{mn}}
\]

(104.412)

\[
\frac{\partial \cos 3\theta}{\partial \sigma_{mn}} = -3 \sqrt{6} \frac{\partial n_{ij}}{\partial \sigma_{mn}} (n_{jk} n_{ki})
\]

(104.413)

\[
\frac{\partial A_d}{\partial \sigma_{mn}} = A_0 z_{ab} \frac{\partial n_{ab}}{\partial \sigma_{mn}} \{ z_{ab} n_{ab} \}
\]

(104.414)

and define \( \{ X \} = 1 \) if \( X > 0 \), and \( \{ X \} = 0 \) if \( X \leq 0 \).
Analytical expression of $\frac{\partial m_{ij}}{\partial \alpha_{kl}}$:

$$\frac{\partial m_{ij}}{\partial \alpha_{mn}} = B \frac{\partial n_{ij}}{\partial \alpha_{mn}} + n_{ij} \frac{\partial B}{\partial \alpha_{mn}} + C \frac{\partial n_{ik}}{\partial \alpha_{mn}} n_{kj} + (n_{ik} n_{kj} - \frac{1}{3} \delta_{ij}) \frac{\partial C}{\partial \alpha_{mn}} + \frac{1}{3} \delta_{ij} \frac{\partial D}{\partial \alpha_{mn}}$$

(104.415)

where

$$\frac{\partial n_{ij}}{\partial \alpha_{mn}} = \frac{p}{|\Lambda|} (n_{ij} n_{mn} - I_{ijmn})$$

(104.416)

$$\frac{\partial D}{\partial \alpha_{mn}} = - \frac{\partial A_d}{\partial \alpha_{mn}} \left( \sqrt{\frac{2}{3}} \alpha^d_\theta - \alpha_{ab} n_{ab} \right) - A_d \left( \sqrt{\frac{2}{3}} \frac{\partial \alpha^d_\theta}{\partial \alpha_{mn}} - n_{mn} - \alpha_{ab} \frac{\partial n_{ab}}{\partial \alpha_{mn}} \right)$$

(104.417)

and

$$\frac{\partial B}{\partial \alpha_{mn}} = \frac{3}{2} \left( \frac{1-c}{c} \right) \left( \frac{\partial g}{\partial \alpha_{mn}} \cos 3\theta + g \frac{\partial \cos 3\theta}{\partial \alpha_{mn}} \right)$$

(104.418)

$$\frac{\partial C}{\partial \alpha_{mn}} = 3 \sqrt{\frac{3}{2}} \left( \frac{1-c}{c} \right) \frac{\partial \alpha^d_\theta}{\partial \alpha_{mn}}$$

(104.419)

$$\frac{\partial \alpha^d_\theta}{\partial \alpha_{mn}} = M_e \exp (n^d \psi) \frac{\partial g}{\partial \alpha_{mn}}$$

(104.420)

$$\frac{\partial g}{\partial \alpha_{mn}} = g^2 \left( \frac{1-c}{2c} \right) \frac{\partial \cos 3\theta}{\partial \alpha_{mn}}$$

(104.421)

$$\frac{\partial \cos 3\theta}{\partial \alpha_{mn}} = -3 \sqrt{6} \frac{\partial m_{ij}}{\partial \alpha_{mn}} (n_{jk} n_{ki})$$

(104.422)

$$\frac{\partial A_d}{\partial \alpha_{mn}} = A_0 z_{ab} \frac{\partial n_{ab}}{\partial \alpha_{mn}} \{z_{ab} n_{ab} \}$$

(104.423)

Analytical expression of $\frac{\partial m_{ij}}{\partial z_{mn}}$:

$$\frac{\partial m_{ij}}{\partial z_{mn}} = \frac{1}{3} \delta_{ij} \frac{\partial D}{\partial z_{mn}}$$

(104.424)

where

$$\frac{\partial D}{\partial z_{mn}} = - \frac{\partial A_d}{\partial z_{mn}} \left( \sqrt{\frac{2}{3}} \alpha^d_\theta - \alpha_{ab} n_{ab} \right)$$

(104.425)
and
\[
\frac{\partial A_d}{\partial z_{mn}} = A_0 n_{mn} \{ z_{ab} n_{ab} \} \tag{104.426}
\]

Analytical expression of \( \frac{\partial A_{ij}}{\partial \sigma_{mn}} \):
\[
\frac{\partial A_{ij}}{\partial \sigma_{mn}} = \frac{2}{3} \left[ \frac{\partial h}{\partial \sigma_{mn}} \left( \sqrt{\frac{2}{3}} \sigma_b^a n_{ij} - \alpha_{ij} \right) + \sqrt{\frac{2}{3}} h \left( n_{ij} \frac{\partial \sigma_b^a}{\partial \sigma_{mn}} + \alpha_{ij} \frac{\partial n_{ij}}{\partial \sigma_{mn}} \right) \right] \tag{104.427}
\]

where
\[
\frac{\partial \alpha_b^a}{\partial \sigma_{mn}} = M_c \exp (-n^{b} \psi) \left( \frac{\partial g}{\partial \sigma_{mn}} - n^{b} g \frac{\partial \psi}{\partial \sigma_{mn}} \right) \tag{104.428}
\]
\[
\frac{\partial h}{\partial \sigma_{mn}} = \frac{1}{(\alpha_{ab} - \alpha_{ab}^m) n_{ab}} \left[ \frac{\partial b_0}{\partial \sigma_{mn}} - h (\alpha_{pq} - \alpha_{pq}^m) n_{pq} \frac{\partial n_{pq}}{\partial \sigma_{mn}} \right] \tag{104.429}
\]

and
\[
\frac{\partial b_0}{\partial \sigma_{mn}} = \frac{b_0}{\delta p} \delta_{mn} \tag{104.430}
\]

Analytical expression of \( \frac{\partial A_{ij}}{\partial \alpha_{mn}} \):
\[
\frac{\partial A_{ij}}{\partial \alpha_{mn}} = \frac{2}{3} \left[ \left( \sqrt{\frac{2}{3}} \sigma_b^a n_{ij} - \alpha_{ij} \right) \frac{\partial h}{\partial \alpha_{mn}} \right.
+ \sqrt{\frac{2}{3}} h \left( n_{ij} \frac{\partial \sigma_b^a}{\partial \alpha_{mn}} + \alpha_{ij} \frac{\partial n_{ij}}{\partial \alpha_{mn}} - I_{ijmn}^\epsilon \right) \right] \tag{104.431}
\]

where
\[
\frac{\partial \alpha_b^a}{\partial \alpha_{mn}} = M_c \exp (-n^{b} \psi) \frac{\partial g}{\partial \alpha_{mn}} \tag{104.432}
\]
\[
\frac{\partial h}{\partial \alpha_{mn}} = -\frac{h}{(\alpha_{ab} - \alpha_{ab}^m) n_{ab}} \left[ n_{mn} + (\alpha_{pq} - \alpha_{pq}^m) n_{pq} \frac{\partial n_{pq}}{\partial \alpha_{mn}} \right] \tag{104.433}
\]

Analytical expression of \( \frac{\partial A_{ij}}{\partial z_{mn}} \):
\[
\frac{\partial A_{ij}}{\partial z_{mn}} = 0 \tag{104.434}
\]

Analytical expression of \( \frac{\partial Z_{ij}}{\partial \sigma_{mn}} \):
\[
\frac{\partial Z_{ij}}{\partial \sigma_{mn}} = -c_z \left[ (z_{max} n_{ij} + z_{ij}) \frac{\partial D}{\partial \sigma_{mn}} + z_{max} D \frac{\partial n_{ij}}{\partial \sigma_{mn}} \right] \{D\} \tag{104.435}
\]

Analytical expression of \( \frac{\partial Z_{ij}}{\partial \alpha_{mn}} \):

\[
\frac{\partial Z_{ij}}{\partial \alpha_{mn}} = -c_z \left[ (z_{max} n_{ij} + z_{ij}) \frac{\partial D}{\partial \alpha_{mn}} + z_{max} D \frac{\partial n_{ij}}{\partial \alpha_{mn}} \right] \{D\} \tag{104.436}
\]

Analytical expression of \( \frac{\partial Z_{ij}}{\partial z_{mn}} \):

\[
\frac{\partial Z_{ij}}{\partial z_{mn}} = -c_z \left( D I_{ijmn}^s + z_{max} n_{ij} \frac{\partial D}{\partial z_{mn}} \right) \{D\} \tag{104.437}
\]

104.6.12 SaniSand2008 (aka SANISAND) Model

Taiebat and Dafalias (2008)

104.6.13 SANICLAY Model

Dafalias et al. (2006)

104.6.14 Pisanò Elastic-Plastic Model with Vanishing Elastic Region (for \( G/G_{\text{max}} \) Modeling)

A more recent description of this model is available by Pisanò and Jeremić (2014).

Modeling the mechanical behavior of soils under cyclic/dynamic loading is crucial in most Geotechnical Earthquake Engineering (GEE) applications, including site response analysis and soil structure interaction (SSI) problems. In the last decades, a number of experimental studies (Ishihara, 1996; di Prisco and Wood, 2012) pointed out the complexity of such behavior – especially in the presence of pore fluid(s) – characterized by non-linearity, irreversibility, anisotropy, barotropy, picnotropy, rate-sensitivity, etc. In principle, a comprehensive soil model should be capable of reproducing all the aspects of the mechanical response for any loading condition, as well as predicting the occurrence of liquefaction and cyclic mobility, distinguishing the conditions for shakedown or ratcheting under repeated loads and so forth. However, such an ideal model would probably require too many data for calibration, along with a cumbersome numerical treatment.

Traditionally, many GEE problems are still tackled in the frequency domain through 1D (equivalent) linear models, mainly because of their computational convenience and straightforward calibration. In
the light of a Kelvin-Voigt visco-elastic idealization, the dynamic soil behavior is fully described in terms of strain-dependent stiffness degradation \( (G/G_{\text{max}}) \) and damping \( (\zeta) \) ratios (Kramer, 1996a). As it holds in the linear regime, the shear and the volumetric responses are assumed to be decoupled, so that \( G/G_{\text{max}} \) and \( \zeta \) curves are derived from the experimental cyclic shear tests (triaxial, simple shear or torsional) as a function of the cyclic shear strain amplitude.

Owing to the availability of computer programs for 1D site response analysis (SHAKE (Schnabel et al., 1972), EERA (Bardet et al., 2000), DEEPSOIL (Hashash and Park, 2001)) and SSI problems (SASSI (Lymser, 1988)), the visco-elastic approach has become more and more popular among practitioners, regardless of drawbacks:

- despite a non-negligible rate-sensitiveness, most energy dissipation in soils derives from frictional inter-granular mechanisms rather than viscous flow (as it is implicitly assumed by using \( G/G_{\text{max}} \) and damping approaches);
- \( G/G_{\text{max}} \) and \( \zeta \) curves do not allow to evaluate irreversible deformations, nor the influence of pore fluid(s);
- adopting 1D shear constitutive relationships has poor mechanical soundness, since soil behavior exhibits a pronounced deviatoric-volumetric multiaxial coupling;
- the meaning of cyclic shear strain amplitude for the choice of \( G/G_{\text{max}} \) and \( \zeta \) values is not evident in the presence of irregular seismic loads.

The above observations justify the need for alternative approaches and more physically consistent soil models. From this standpoint, the incremental elastic-plastic theory represents the main modeling framework, within which significant efforts have been spent in the last decades to simulate the response of cyclically loaded soils. Several approaches have been explored and gradually refined, including e.g. “multi-surface plasticity”, “bounding surface plasticity”, “generalized plasticity” and “hypoplasticity”. Comprehensive overviews on cyclic elasto-plasticity modeling can be found in Lemaitre and Chaboche (1990) and, with specific reference to soils, Prevost and Popescu (1996), Zienkiewicz et al. (1999b) and di Prisco and Wood (2012). In most cases, rotational kinematic hardening formulations have been adopted in conjunction with increasingly accurate flow rules and hardening laws; a number of valuable contributions are worth citing, such as – to mention only a few – Mróz et al. (1978); Prevost (1985a); Borja and Amies (1994); Manzari and Dafalias (1997); Papadimitriou and Bouckovalas (2002); Elgamal et al. (2002); Taiebat and Dafalias (2008); recently, it has been also shown how a good simulation of dynamic properties can be achieved by means of even elastic-perfectly plastic models, as long as formulated in a probabilistic elastic-plastic framework (Sett et al., 2011b). The major issues about the
practical use of elastic-plastic models concern the complexity of the mathematical formulations, the computer implementation and the possible high number of material parameters. For a model to appeal to practicing engineers, a tradeoff is needed between the overall accuracy and the number of parameters to be calibrated, particularly provided the frequent lack of detailed in situ or laboratory data.

Among the aforementioned models, the one by Borja and Amies (1994) is here taken into special consideration. These authors proposed a total-stress von Mises-type model in the framework of kinematic-hardening bounding surface plasticity, then successfully applied to the seismic simulation of fine-grained deposits at Lotung site in Taiwan (Borja et al., 1999, 2000). Based on work by Dafalias and Popov (1977) and Dafalias (1979), the multiaxial model is characterized by the assumption of vanishing elastic domain, thus implying soil plastification under any load level and a redefinition of the standard loading/unloading criterion. Apart from the mathematical aspects, the model possesses sufficient flexibility to reproduce the undrained dynamic properties of clayey/silty soils, while keeping a minimum number of physically-based parameters.

In this paper similar bounding surface approach with vanishing elastic region is adopted to derive a Drucker-Prager effective-stress model, incorporating pressure sensitivity and non-associativeness, essential ingredients for material modeling of granular materials. As a result, the following constitutive relationship is suitable for the effective-stress time-domain analysis of even liquefiable soils. In addition, the dissipative model performance is here explored in combination with a further viscous mechanism, which can be wisely exploited to improve the simulation of the experimental damping. Although numerical convenience often motivates the embedment of viscous dissipation into elastic-plastic computations, it has a de facto physical origin, coming from rate-dependent processes occurring at both inter-granular contacts and grain/pore fluid interfaces.

104.6.14.1 Frictional and viscous dissipative mechanisms

The time-domain finite element (FE) solution of dynamic problems is usually carried out by solving an incremental discrete system of the following form (Bathe, 1982; Zienkiewicz and Taylor, 1991a):

\[ M\Delta \ddot{U} + C\Delta \dot{U} + K^t\Delta U = \Delta F^{ext} \]  (104.438)

where \( \Delta \) and dots stand respectively for step increment and time derivative, \( U \) is the generalized DOF vector (nodal displacement for example), \( F^{ext} \) the nodal external force vector and \( M, C, K^t \) are the mass, damping and (tangent) stiffness matrices, respectively.

In system (104.438) two dissipative sources are readily recognizable, namely the viscous (velocity-proportional) and the frictional (displacement-proportional) terms (Argyris and Mlejnek, 1991). While the latter is given by the variation of the elastic-plastic tangent stiffness \( K^t \), the viscous term related
to the damping matrix $C$ can represent interaction of solid skeleton and pore fluid, and constitutive
time-sensitiveness of the soil skeleton. The above combination of frictional and viscous dissipation can
be interpreted in terms of two distinct effective stress components acting on the soil skeleton:

$$\sigma_{ij} = \sigma_{ij}^f + \sigma_{ij}^v$$

(104.439)

where the effective stress tensor $\sigma_{ij}$ has been split into frictional (elastic-plastic) and viscous stresses\(^{66}\).

From a rheological point of view, the resulting scheme can be defined as visco-elastic-plastic – not elastic-
viscoplastic – as the elastic-plastic response is rate-independent and accompanied by a parallel viscous
resisting mechanism. In what follows, the frictional component is first specified via the formulation of
the bounding surface model with vanishing elastic domain; then, the role and the calibration of the linear
viscous term is discussed.

Index tensor notation is used, along with the standard Einstein convention for repeated indices; the
norm of any second-order tensor $x_{ij}$ is defined as $\|x_{ij}\| = \sqrt{x_{ij}x_{ij}}$, whereas the deviatoric component
can be extracted as $x_{ij}^{dev} = x_{ij} - x_{kk}\delta_{ij}/3$ ($\delta_{ij}$ is the Kronecker delta). In accordance with usual Solid
Mechanics conventions, positive tensile stresses/strains are considered, whereas – as is done in Fluid
Mechanics – only the isotropic mean pressure is positive if compressive.

### 104.6.14.2 Bounding surface frictional model with vanishing elastic domain

The formulated constitutive model represents the frictional effective-stress version of the previous work
by Borja and Amies (1994); for the sake of clarity, the presentation sequence of the former publication
is here maintained, highlighting both differences and similarities. As was expected, the introduction
of pressure-dependence into the constitutive equations implies somewhat more involved derivations, so
that the analytical details skipped in this section are reported toward the end in section 104.6.14.7; the
superscript $f$ referring to the frictional component of the global effective stress (Equation (104.439)) is
avoided for the sake of brevity.

**Elastic relationship** Provided the usual additive (incremental) strain split into elastic and plastic
components $d\varepsilon_{ij} = d\varepsilon_{ij}^e + d\varepsilon_{ij}^p$, the incremental linear elastic Hooke’s law is expressed as follows:

$$d\sigma_{ij} = D_{ijhk}^e \left( d\varepsilon_{ehk} - d\varepsilon_{ihk}^p \right)$$

(104.440)

where $d$ stands for a differentially small increment of strain and $D_{ijhk}^e$ is the fourth-order elastic stiffness
tensor. Under the elastic deviatoric/volumetric decoupling, the deviatoric and volumetric counterparts

\(^{66}\)Henceforth, effective stresses are exclusively accounted for
of Equation (104.440) can be also given:

\[
ds_{ij} = 2G_{\text{max}} (d\epsilon_{hk} - d\epsilon_{hk}^p) \tag{104.441}
\]

\[
dp = -K (d\epsilon_{\text{vol}} - d\epsilon_{\text{vol}}^p) \tag{104.442}
\]

in which \( p = -\sigma_{kk}/3 \) is the mean stress, \( \epsilon_{\text{vol}} = \epsilon_{kk} \) is the volumetric strain, \( s_{ij} = \sigma_{ij}^{\text{dev}} \) is the stress deviator, and \( e_{ij} = \epsilon_{ij}^{\text{dev}} \) is the strain deviator. The shear modulus \( G_{\text{max}} = E/(2(1+\nu)) \) and the bulk modulus \( K = E/(3(1-2\nu)) \) are derived from the Young modulus \( E \) and the Poisson’s ratio \( \nu \). Henceforth, \( G_{\text{max}} \) will be always used for the elastic small-strain shear modulus, whereas the secant cyclic shear stiffness will be referred to as \( G \).

**Drucker-Prager yield and bounding loci**  A conical Drucker-Prager type yield locus is first introduced, similar to what is used by Prevost (1985a) and Manzari and Dafalias (1997):

\[
f_y = \frac{3}{2} (s_{ij} - p\alpha_{ij}) (s_{ij} - p\alpha_{ij}) - k^2 p^2 = 0 \tag{104.443}
\]

where \( \alpha_{ij} \) is the so-called deviatoric back-stress ratio \( (\alpha_{kk} = 0) \) governing the kinematic hardening of the yield surface; \( k \) is a parameter determining the opening angle of the cone. It is also important to note that the variation of the back-stress ratio \( \alpha_{ij} \) in (104.443) determines a rigid rotation of the yield locus and, therefore, a rotational kinematic hardening.

The stress derivative of the yield function is also reported for the following developments:

\[
\frac{\partial f_y}{\partial \sigma_{ij}} = \left( \frac{\partial f_y}{\partial \sigma_{ij}} \right)^{\text{dev}} + \left( \frac{\partial f_y}{\partial \sigma_{ij}} \right)^{\text{vol}} = 3 (s_{ij} - p\alpha_{ij}) + \left[ \alpha_{hk} (s_{hk} - p\alpha_{hk}) + \frac{2}{3} k^2 p \right] \delta_{ij} \tag{104.444}
\]

The yield locus must always reside within the so-called bounding surface, here assumed to be a further Drucker-Prager cone (non kinematically hardening, fixed in size):

\[
f_B = \frac{3}{2} s_{ij} s_{ij} - M^2 p^2 = 0 \tag{104.445}
\]

where \( M \) provides the bounding cone opening and, as a consequence, the material shear strength (as a function of the mean effective pressure \( p \)).
Plastic flow and translation rule  When dealing with granular materials, a non-associated plastic flow rule is needed (Nova and Wood, 1979), allowing for plastic contractancy or dilatancy depending on whether loose or dense materials are analyzed. Here, the plastic flow rule is borrowed from Manzari and Dafalias (1997):

\[ \frac{d\epsilon_p}{dhk} = d\lambda \left( n_{ij}^{dev} - \frac{1}{3} D\delta_{ij} \right) \quad \text{(104.446)} \]

where \( d\lambda \) is the plastic multiplier, \( n_{ij}^{dev} \) is a deviatoric unit tensor (\( \| n_{ij}^{dev} \| = 1 \)) and \( D \) is a dilatancy coefficient defined as (Manzari and Dafalias, 1997):

\[ D = \xi \left( \alpha_{ij}^d - \alpha_{ij} \right) n_{ij}^{dev} = \xi \left( \sqrt{\frac{2}{3}} k_d n_{ij}^{dev} - \alpha_{ij} \right) n_{ij}^{dev} \quad \text{(104.447)} \]

in which \( \xi \) and \( k_d \) are two positive constitutive parameters. While the former controls the amount of volumetric plastic strain, the latter determines the position of the so called “dilatancy surface” and rules the transition from contractive \( (D > 0) \) to dilative \( (D < 0) \) behavior.

The kinematic hardening evolution of the yield locus is imposed via the standard Prager translation rule for the (deviatoric) back-stress ratio (Borja and Amies, 1994):

\[ d\alpha_{ij} = \| d\alpha_{ij} \| n_{ij}^{dev} \quad \text{(104.448)} \]

with both \( n_{ij}^{dev} \) and the norm \( \| d\alpha_{ij} \| \) to be determined.

Vanishing elastic region and consistency condition  As previously mentioned, the most notable feature of the present model concerns the vanishing elastic domain, corresponding with the limit \( k \to 0 \) in Equation (104.443). Accordingly, the Drucker-Prager cone reduces to its symmetry axis, so that:

\[ \lim_{k \to 0} f_y = 0 \Rightarrow \lim_{k \to 0} s_{ij} = p\alpha_{ij} \Rightarrow ds_{ij} = d\alpha_{ij}p + \alpha_{ij}dp \quad \text{(104.449)} \]

and, after substituting the Prager rule (104.448) (for more detailed derivation, see section 104.6.14.7):

\[ n_{ij}^{dev} = \frac{ds_{ij} - \alpha_{ij}dp}{\| ds_{ij} - \alpha_{ij}dp \|} \quad \text{(104.450)} \]

The direction of the deviatoric plastic strain increment \( n_{ij}^{dev} \) depends on the variation of both the stress deviator and the mean pressure, which differs from the cohesive version by Borja and Amies (1994). It
is also worth noting that purely hydrostatic stress increments ($d s_{ij} = 0$) from initial hydrostatic states ($\alpha_{ij} = 0$) yield $n_{ij}^{\text{dev}} = 0$ and thus generates no deviatoric plastic strains.

From a theoretical standpoint, since the direction of the deviatoric plastic strain increment $n_{ij}^{\text{dev}}$ depends on $d \sigma_{ij}$, the resulting constitutive formulation can be properly defined as “hypoplastic”, this being a spontaneous outcome of the limit operation applied on the elastic region (Dafalias, 1986).

The norm of the back-stress increment in Equation (104.448) is obtained by imposing the standard consistency condition, that is the fulfillment of $d f_y = 0$ during plastic loading (section 104.6.14.7):

$$
 df_y = 0 \iff \frac{\partial f}{\partial \sigma_{ij}} d \sigma_{ij} + \frac{\partial f}{\partial \alpha_{ij}} d \alpha_{ij} = 0
$$

whence:

$$
 ||d \alpha_{ij}|| = \frac{1}{p N^\text{dev}} \frac{\partial f}{\partial \sigma_{ij}} d \sigma_{ij}
$$

and $N^\text{dev} = ||(d f_y/\partial \sigma_{ij})^{\text{dev}}|| = 3 ||s_{ij} - p \alpha_{ij}||$. From Equation (104.452), the norm of $d \alpha_{ij}$ can be further specified for the case of radial loading paths in the deviatoric plane, characterized by the nullity of $dp$ and the coaxiality of $s_{ij}$, $\alpha_{ij}$ and their increments. After simple manipulations (see section 104.6.14.7) this results in:

$$
 ||d \alpha_{ij}|| = \sqrt{\frac{2}{3}} \frac{d q}{p}
$$

where $q = \sqrt{3/2} ||s_{ij}||$ stands for the usual deviatoric stress invariant.

**Hardening relationship and plastic multiplier**  An incremental hardening relationship is directly established (Borja and Amies, 1994):

$$
 dq = \sqrt{\frac{2}{3}} H \|d e_{ij}^{p}\|
$$

where $H$ is the hardening modulus. Then, the substitution of both the flow rule (104.446) and the hardening relationship (104.454) into (104.453) leads to:

$$
 ||d \alpha_{ij}|| = \frac{2}{3} H d \lambda \frac{1}{p}
$$
By equaling the right–hand sides of Equations (104.441) and (104.449) the following relationship is obtained:

\[
2G_{\text{max}} \left( d_{ij} - d\lambda n_{ij}^{\text{dev}} \right) = \| d\alpha_{ij} \| s_{ij}^{\text{dev}} p + \alpha_{ij} dp = \frac{2}{3} \frac{H d\lambda}{p} n_{ij}^{\text{dev}} p - \alpha_{ij} K (d_{\text{vol}} + d\lambda D) \tag{104.456}
\]

whence:

\[
d\lambda = \frac{2G_{\text{max}} \| d_{ij} \| + K d_{\text{vol}} \alpha_{ij} n_{ij}^{\text{dev}}}{2G + \frac{2}{3} H - K D \alpha_{ij} n_{ij}^{\text{dev}}} \tag{104.457}
\]

Equation (104.457) represents the consistent "frictional" generalization of Equation (18) in Borja and Amies (1994), as well as the limit of Equation (12) in Manzari and Dafalias (1997)\textsuperscript{67} for a vanishing yield locus size.

**Stress projection, hardening modulus and unloading criterion**  The bounding surface plasticity theory relies on the basic concept that the plastic modulus explicitly depends on the distance between the current stress state and an *ad hoc* stress projection onto the bounding surface. While Borja and Amies (1994) defined a purely deviatoric projection operator, here the whole stress state is involved:

\[
\bar{\sigma}_{ij} = \sigma_{ij} + \beta (\sigma_{ij} - \sigma_{ij}^0) \tag{104.458}
\]

where \( \beta \) is a scalar distance coefficient and \( \sigma_{ij}^0 \) embodies the stress state at the last stress reversal (Figure 104.13). The coefficient \( \beta \) must be such that the projected stress \( \bar{\sigma}_{ij} \) lies on the bounding surface (Equation (104.445)):

\[
\frac{3}{2} \bar{s}_{ij} s_{ij} = M^2 p^2 \tag{104.459}
\]

whence, after substituting (104.458) into (104.459), \( \beta \) can be obtained as the positive root of the following second-order algebraic equation:

\[
\frac{3}{2} \left[ s_{ij} - s_{ij}^0 \right]^2 - \frac{2}{3} M^2 (p - p_0)^2 \right] \beta^2 + \frac{2}{3} M^2 p (p - p_0) \right] \beta + \left[ \| s_{ij} \|^2 - \frac{2}{3} M^2 p^2 \right] = 0 \tag{104.460}
\]

\textsuperscript{67}Different signs result because of the opposite sign conventions adopted by these authors
Apparently, the limit situations $\beta = 0$ and $\beta \to \infty$ correspond with the current stress state being right on the bounding locus or at instantaneous unloading (stress reversal).

In principle, any analytical relationship can be adopted to relate $H$ and $\beta$, as long as two fundamental requirements are satisfied, i.e. $H (\beta = 0) = 0$ and $H (\beta \to \infty) \to \infty$: the former ensures the material shear strength to be fully mobilized when the bounding surface is attained; the latter guarantees an instantaneous elastic stiffness upon any stress reversal, as is explained next. In this case, the expression by Borja and Amies (1994) has been extended to frictional media by accounting for the influence of the mean pressure:

$$H = ph\beta^m$$

in which $h$ and $m$ are two additional constitutive parameters.

The last element of the model formulation is the unloading criterion, which in this case is ill-defined due to the lack of the elastic region and the yield surface. The same multiaxial unloading criterion suggested by Borja and Amies (1994) is employed, based on the observation that the hardening modulus
$H$ increases at the onset of unloading. Accordingly, as long as $H(\beta)$ is a monotonically increasing function, instantaneous unloading is assumed to occur whenever $dH > 0$, i.e. $d\beta > 0$. The variation of $\beta$ (and thus its sign) can be readily derived by substituting (104.458) into (104.459), and then differentiating the latter with respect to $\beta$:

$$d\beta = -(1 + \beta) \frac{\bar{s}_{ij} ds_{ij} - \frac{2}{3} M^2 \bar{p} dp}{\bar{s}_{ij} (s_{ij} - s_{ij}^0) - \frac{2}{3} M^2 \bar{p} (p - p^0)} > 0$$

(104.462)

It is worth noting that the variation of $\beta$, $d\beta$, plays here the same role of the scalar product $(\partial f/\sigma_{ij}) d\sigma_{ij}$ in standard elastic-plastic models, i.e. it defines the alternatives of elastic-plastic loading ($d\beta < 0$), neutral loading ($d\beta = 0$) or elastic unloading ($d\beta > 0$). The last key point concerns the update of the stress $\sigma_{ij}^0$ in Equation (104.458), which must be set equal to the current stress state when $d\beta > 0$ is instantaneously found.

**Possible refinements** The frictional model has been developed trying to keep the number of material parameters as low as possible, even with a non-associated flow rule. However, it is worth mentioning which kind of improvements might be introduced if required by the problem under examination.

It should be first noted that, as a Drucker-Prager type bounding surface has been adopted, the material shear strength is unaffected by the Lode angle, so that for instance the same failure obliquity is predicted for triaxial compression and extension. This drawback could be easily remedied by modifying the deviatoric cross-section of the bounding surface itself, e.g. by adopting the well known Mohr-Coulomb deviatoric locus or other smooth loci (Matsuoka and Nakai, 1974; Willam and Warnke, 1974; Lade, 1977). A change in the deviatoric cross-section would negligibly influence the overall formulation, as just the evaluation of the projection distance $\beta$ and of its increment should be modified (Equations (104.459)-(104.460) and (104.462)).

Secondly, the present version of the model cannot predict a possible brittle behavior of the soil, usually taking place in the case of dense materials. Constitutive brittleness could be accounted for by incorporating a further isotropic hardening mechanism at the bounding surface level, allowing for a gradual shrinkage of the outer surface during plastifications.

Another relevant point is about the fact that different parameters must be calibrated for different relative densities of the same granular material, as if distinct materials were indeed considered. As a matter of fact, continuous transitions from loose to dense states (and vice versa) spontaneously take place during straining: this aspect has been successfully addressed and reproduced via the concept of "state parameter" (Been and Jefferies, 1985; Wood et al., 1994; Manzari and Dafalias, 1997), which could be also introduced into a critical-state version of the proposed model.
The above and further refinements – related for instance to non-linear elastic behavior, anisotropy, fabric effects, delayed plastic response, etc. – might result in a more accurate soil model, implying though higher difficulties in terms of calibration, implementation and, as a consequence, practical employment.

104.6.14.3 The role of linear viscous damping

An additional viscous mechanism (Equation (104.439)) can be usually exploited in finite element (FE) analysis, even though it is not directly included in the constitutive model. Indeed, many numerical codes solve discrete systems with a viscous damping term (Equation (104.438)), usually assembled as a linear combination of the mass and the (elastic) stiffness matrices (Rayleigh formulation (Argyris and Mlejnek, 1991; Chopra, 2000)):

$$C = a_0 M + a_1 K^e$$  (104.463)

where \(a_0\) and \(a_1\) are two constant parameters, related to the \(n^{th}\) modal damping ratio \(\zeta_n\) of the discrete structural system.

It could be easily shown that a constitutive viscosity of the form:

$$\sigma^v_{ij} = D^v_{ijkl} \epsilon_{lk}$$  (104.464)

gives rise to a stiffness-proportional damping matrix, which can be equivalently reproduced through the following calibration of the Rayleigh damping parameters (Borja et al., 2000; Hashash and Park, 2002):

$$a_0 = 0 \quad a_1 = \frac{2\zeta_0}{\omega}$$  (104.465)

The calibration (104.465) establishes the same ratio between tangential/bulk elastic and the viscous moduli, that is \(G^e_{max}/G^v_{max} = K^e/K^v\). More importantly, a damping ratio \(\zeta_0\) is ensured for a given circular frequency \(\omega\), as long as the parallel resisting mechanism \(\sigma^f_{ij}\) is purely elastic; as a consequence, provided the \(a_1\) value at the beginning of the analysis, modal frequencies and the corresponding damping ratios are linearly related.

It is also worth remarking some further points about the implications of linear viscous damping in conjunction with non-linear soil models. If a soil element undergoes an imposed shear strain history, the overall shear stress/strain cycles \(\tau - \gamma\) differ from the purely frictional component \(\tau^f - \gamma\), this difference being due to the viscous shear stress \(\tau^v\). As will be shown in next section, the viscous component implies smoother cycles and avoid the sharp transitions at stress reversal usually exhibited by purely
elastic-plastic responses (Borja et al., 2000). However, the overall $G/G_{\text{max}}$ ratio between the average cyclic stiffness and the elastic shear modulus is unaffected by viscosity.

As far as the damping ratio is concerned, its standard definition (Kramer, 1996a) can be easily adapted to point out the frictional/viscous splitting of the energy $\Delta W$ dissipated in a loading cycle:

$$\zeta = \frac{\Delta W}{2\pi G\gamma^2_{\text{max}}} = \frac{\Delta W^f + \Delta W^v}{2\pi G\gamma^2_{\text{max}}} = \zeta^f + \zeta^v$$

(104.466)

where $\gamma_{\text{max}}$ is the imposed cyclic shear strain amplitude and $G$ the corresponding (secant) cyclic shear stiffness. As $\gamma_{\text{max}}$ approaches zero, the plastic dissipation tends to zero as well, so that $\zeta = \zeta^v$; therefore, the Rayleigh parameter $a_1$ can be calibrated to obtain $\zeta (\gamma_{\text{max}} \to 0) = \zeta_0$ for a given circular frequency $\omega$ (see Equation (104.465)). This is a desirable feature of the model, as natural soils are well known to dissipate energy at even very small strain amplitudes.

At progressively larger strains, both the frictional and viscous components contribute to the global damping, although the relative quantitative significance is hard to assess a priori. In addition, the viscous component of the $\zeta - \gamma_{\text{max}}$ curve is not constant, since $\zeta^v$ depends on the strain-dependent secant modulus $G(\gamma_{\text{max}})$ and, implicitly, on the strain rate. This is different to what has been argued by Borja et al. (2000).

As an example, consider the response of an elastic-perfectly plastic model with additional viscosity under a sinusoidal shear excitation $\gamma(t) = \gamma_{\text{max}} \sin(\omega t)$. The simplicity of the elastic-perfectly plastic response allows derivation of instructive analytical formulas for the $G/G_{\text{max}}$ and the damping ratios, even in the presence of viscous dissipation. While $\gamma_{\text{max}} < \gamma_y$ (yielding shear strain), the material behavior is linear elastic, so that $G/G_{\text{max}} = 1$ and $\zeta$ equals the purely viscous contribution at $\gamma_{\text{max}} = 0$, i.e. $\zeta = \zeta_0$; $\gamma_y$ depends on the elastic stiffness and the shear strength of the material, $\gamma_y = \tau_{\text{lim}}/G_{\text{max}}$, where $\tau_{\text{lim}}$ is the limit (frictional) shear stress for a given confining pressure. For $\gamma_{\text{max}} > \gamma_y$ plastifications take place with a flat elastic-perfectly plastic $\tau^f - \gamma$ branch, and the following expressions can be easily derived:

$$\frac{G}{G_{\text{max}}} = \frac{\tau_{\text{lim}}}{G_{\text{max}}\gamma_{\text{max}}}$$

(104.467)

$$\zeta = \frac{\Delta W^f + \Delta W^v}{2\pi G\gamma^2_{\text{max}}} = \frac{2}{\pi} \left( 1 - \frac{\tau_{\text{lim}}}{G_{\text{max}}\gamma_{\text{max}}} \right) + \zeta_0 \frac{G_{\text{max}}\gamma_{\text{max}}}{\tau_{\text{lim}}}$$

(104.468)

In Figure 104.14 the $G/G_{\text{max}}$ and $\zeta$ ratios are plotted for increasing $\zeta_0$ values. As $\gamma_{\text{max}}$ increases, the frictional damping tends to $2/\pi \approx 0.63$, while the viscous one keeps increasing because of the reduction in
the secant stiffness and the increase in the shear strain rate (depending on the strain amplitude). Hence, the value of $\zeta_0$ is to be carefully chosen, in order to avoid excessive dissipation when medium/large strains are induced by the loading process.

![Graph](image.png)

Figure 104.14: $G/G_{max}$ and damping curves for a elastic-perfectly plastic model with linear viscous damping at varying $\zeta_0$ ($\tau_{lim}=100$ kPa, $G_{max}=100$ MPa)

The fact that the viscous mechanism can modify the purely frictional $\zeta - \gamma_{max}$ curve without altering the cyclic stiffness degradation can be fruitfully exploited to remedy the (frequent) cases in which the experimental-numerical agreement is not satisfactory in terms of energy dissipation properties.

104.6.14.4 Model performance and calibration

The frictional mechanism of the above model is characterized by a rather low number of material parameters, namely the following seven:

- two elastic parameters, the Young modulus $E$ (or the shear modulus $G_{max}$) and the Poisson’s ratio $\nu$;
- the shear strength parameter $M$ for the definition of the bounding surface (Equation (104.445));
- the flow rule parameters, $\xi$ and $k_d$, governing the increment of the volumetric plastic strain under shearing and the size of the dilatancy surface, respectively (Equation (104.446));
- the hardening parameters $h$ and $m$ for the dependence of the hardening modulus on the distance coefficient $\beta$ (Equation (104.461)), affecting the pre-failure deformatonal behavior and, in overall, the resulting dynamic properties ($G/G_{max}$ and damping curves).
Provided a reasonable value for the Poisson’s ratio (usually in the range $0.25 - 0.3$), the small-strain elastic stiffness can be evaluated from dynamic laboratory (RC tests) or in situ (seismic geophysical surveys) tests. As far as the shear strength is concerned, the parameter $M$ can be related to the friction angle $\phi$ as follows:

$$M = \frac{6 \sin \phi}{3 \pm \sin \phi}$$

(104.469)

depending on whether triaxial compression (sign $-$ in (104.469)) or extension (sign $+$ in (104.469)) failure conditions are to be reproduced (a change in the deviatoric section of the bounding surface would allow to capture both compressive and extensive limits).

The calibration of the flow rule parameters, $\xi$ and $k_d$, requires at least a triaxial test to be performed, in order to obtain some information about the volumetric behavior. Figure 104.15 shows the predicted triaxial response for three different values of $k_d$ (and fixed $\xi$), that is by varying the opening angle of the dilatancy surface (the employed parameters are reported in the figure caption, where $p_0$ stands for the initial mean pressure).

While the limit stress deviator $q$ is exclusively given by $M$, the pre-failure behavior is influenced by the plastic deformability and therefore by $k_d$. The model possesses sufficient flexibility to reproduce contractive, dilative or contractive/dilative behavior; also, such a feature is necessary to reproduce undrained conditions (liquefying and non-liquefying responses), this being a further motivation for non-associativeness when dealing with sandy materials.

Figure 104.16 exemplifies the response predicted under pure shear (PS) cyclic loading, applied as a sinusoidal shear strain history ($\gamma_{\text{max}} = 0.2\%, 20\%$, period $T=2\pi$ s) at constant normal stresses (and thus constant mean pressure $p_0$ as well. This corresponds with a radial loading path on the deviatoric plane); for the sake of clarity, the volumetric plastic response has been inhibited ($\xi=0$), in order to evaluate the deviatoric mechanism exclusively. Both purely frictional (solid line) and frictional/viscous (dashed line) responses are plotted.

Owing to the kinematic hardening of the vanished yield locus, the model is capable of reproducing both the Bauschinger and the Masing effects, the latter implying the stabilization of the cyclic response to take place after more than one loading cycle. As expected, the additional viscous damping increases the area of the cyclic loop and therefore the overall dissipated energy; however, the effect of the viscous dissipation becomes significant only at medium-high shear strains, corresponding – for a given loading frequency – with higher strain rates. Further, viscosity causes the aforementioned “smoothing” of stress reversals, as it can be readily noticed in Figure 104.17 by comparing the purely frictional and the frictional/viscous responses.

Given the elastic stiffness and the strength of the soil, the shape of the resulting loading cycles is
totally governed by the hardening properties, by $h$ and $m$ in Equation (104.461): this directly affects the simulation of experimental $G/G_{\text{max}}$ and damping curves, which can be therefore exploited for the calibration of both $h$ and $m$. As can be easily demonstrated (the proof is given in section 104.6.14.7), the following equality holds under PS loading conditions, i.e. under constant pressure shearing:

$$1 = \frac{G}{G_{\text{max}}} \left[ 1 + \frac{6G_{\text{max}}}{hp_0\gamma_{\text{max}}} \int_0^{\gamma_{\text{max}}} \left( \eta_{\text{lim}}/G - 2\gamma + \gamma_{\text{max}} \right)^m d\gamma \right]$$

(104.470)

where $\eta_{\text{lim}} = M_{p0}/\sqrt{3}$. Relationship (104.470) has been obtained by integrating the constitutive equations over the first loading cycle, and represents the frictional counterpart of Equation (6) in Borja et al. (2000) – as is testified by the explicit influence of the confining pressure $p_0$. The proper use
Figure 104.16: Predicted pure shear response at two different shear strain amplitudes ($p_0=100$ kPa, $T=2\pi$ s, $\zeta_0 = 0.003$, $G_{\text{max}}=4$ MPa, $\nu=0.25$, $M=1.2$, $k_d=\xi=0$, $h=G/(1.5p_0)$, $m=1$)

of Equation (104.470) requires first the choice of two meaningful points on the $G/G_{\text{max}}$ experimental curve, i.e. two $(\gamma_{\text{max}},G/G_{\text{max}})$ couples; then, the unknowns $h$ and $m$ are obtained by solving the integral system arising from the specification of Equation (104.470) for both selected $(\gamma_{\text{max}},G/G_{\text{max}})$ couples.

Figure 104.18 illustrates the result of the above calibration procedure, applied on the $G/G_{\text{max}}$ and $\zeta$ curves for sands implemented into the code EERA (Bardet et al., 2000) and formerly obtained by Seed and Idriss (1970b).

Since Equation (104.470) exclusively accounts for the $G/G_{\text{max}}$ curve, the very satisfactory agreement in terms of stiffness degradation (viscosity has no effect on it) should not surprise. On the other hand, once $h$ and $m$ are set, the predicted damping curve may or may not match the experimental outcome.
Figure 104.17: Detail of stress reversals for the pure shear response in Figure 104.16 ($\gamma_{\text{max}} = 20\%$)

Figure 104.18: Comparison between experimental and simulated $G/G_{\text{max}}$ and damping curves ($p_0=100$ kPa, $T=2\pi$ s, $\zeta = 0.003$, $G_{\text{max}} = 4$ MPa, $\nu=0.25$, $M=1.2$, $k_d=\xi=0$, $h=G/(112p_0)$, $m=1.38$) irrespective of the calibration procedure. In this respect, Figure 104.18 also presents the comparison between the damping curve by Seed and Idriss and the model prediction. The frictional $\zeta$ curve lies in the same experimental range, even though the accuracy at $\gamma_{\text{max}} = 0.03 - 1\%$ is not as good as for the $G/G_{\text{max}}$ ratio. In this case, the contribution of the viscous mechanism is practically non-existent, as it only increases the total $\zeta$ ratio for $\gamma_{\text{max}} > 0.1\%$.

Depending on the specific application, a “trial and error” calibration might be preferable, sacrificing some of the accuracy in terms of $G/G_{\text{max}}$ ratio to improve the damping performance. A possible
outcome of a manual calibration is plotted in Figure 104.19: apparently, while the simulation of the stiffness curve is still acceptable, the damping curve appears to be much better than the previous one. The use of the viscous mechanism seems to be highly beneficial, since it remedies the lack of accuracy in the frictional curve at medium/large cyclic strains.

![Comparison between experimental and simulated G/G\textsubscript{max} and damping curves](image)

Figure 104.19: Comparison between experimental and simulated $G/G_{\text{max}}$ and damping curves ($p_0=100$ kPa, $T=2\pi$ s, $\zeta=0.003$, $G_{\text{max}}=4$ MPa, $\nu=0.25$, $M=1.2$, $k_d=\xi=0$, $h=G_{\text{max}}/(15p_0)$, $m=1$)

It is also worth noting that the experimental/numerical agreement is good up to $\gamma_{\text{max}}=10\%$, this being a rather high cyclic strain level, for equivalent elastic modeling of soil. In fact, even though the interpretation of experimental cyclic tests is questionable when substantial plasticity occurs, the proposed model produces, within a different framework, the same mechanical response incorporated into traditional equivalent-linear approaches. Besides, if the experimental data under examination are unsatisfactorily reproduced for any $h$ and $m$ combination, the user still has the chance of substituting the interpolation function (104.461) with no further changes in the model formulation.

104.6.14.5 Parametric analysis

In this section the sensitivity of the model predictions to some relevant input parameters is parametrically investigated.

**Influence of the confining pressure** Figure 104.20 illustrates the sensitivity, under PS loading, of both $G/G_{\text{max}}$ and damping frictional curves to the initial confining pressure. As can be noticed, increasing $p_0$ does enlarge the "pseudo-elastic" range, that is the strain interval within which the deviation by the
elastic behavior is negligible even with a vanishing yield locus. It is also noted that the variations in the confining pressure do not imply appreciable changes in the shape of the curves.

![Graph](image-url)

Figure 104.20: Simulated $G/G_{\text{max}}$ and damping curves at varying confining pressure ($T=2\pi$ s, $G_{\text{max}}=4$ MPa, $\nu=0.25$, $M=1.2$, $k_d=\xi=0$, $h=G/(15p_0)$, $m=1$)

**Influence of the hardening parameters** Figures 104.21 and 104.22 show the influence of the hardening parameters $h$ and $m$ on the predicted cyclic properties. As the material strain-hardening is accelerated by decreasing either $h$ or $m$, the pseudo-elastic range tends to disappear, so that $G/G_{\text{max}} < 1$ and $\zeta > 0$ at even $\gamma_{\text{max}} = 10^{-4}\%$; conversely, an extended pseudo-elastic behavior can be obtained over a large strain range by increasing the hardening parameters. Apparently, the model ensures high flexibility in terms of cyclic curve shapes, so that the response of standard elastic-plastic models (i.e. with non-vanishing elastic region) can be smoothly approximated (compare for instance the $m=3$ curves in Figure 104.22 and the analytical elastic-perfectly plastic frictional curves in Figure 104.14).

**Influence of the viscous mechanism** The influence of the viscous parameter $\zeta_0$ on the resulting frictional/viscous damping curve is illustrated in Figure 104.23 (the $G/G_{\text{max}}$ is not affected by the parallel viscous mechanism). As was expected, an increase in $\zeta_0$ induce larger values of $\zeta (\gamma_{\text{max}} \rightarrow 0)$, as well as a faster increase of the $\zeta$ curve at medium/high cyclic strains. Figure 104.23 confirms the suitability of the viscous mechanism, as an additional degree of freedom for reproducing the cyclic dissipative soil behavior.
Figure 104.21: Simulated $G/G_{\text{max}}$ and damping curves at varying $h$ ($p_0=100$ kPa, $T=2\pi$ s, $G_{\text{max}} = 4$ MPa, $\nu=0.25$, $M=1.2$, $k_d=\xi=0$, $m=1$)

Figure 104.22: Simulated $G/G_{\text{max}}$ and damping curves at varying $m$ ($p_0=100$ kPa, $T=2\pi$ s, $G_{\text{max}} = 4$ MPa, $\nu=0.25$, $M=1.2$, $k_d=\xi=0$, $h=G_{\text{max}}/(15p_0)$)

**Influence of the volumetric behavior in constrained problems** So far, all the simulations have been performed by inhibiting the elastic-plastic soil dilatancy ($\xi = 0$), which in most cases cannot be done to represent real soil behavior. As previously shown for triaxial loading conditions (Figure 104.15), in the absence of kinematic boundary constraints, a variation in the volumetric behavior slightly affects only the hardening evolution of the stress-strain response toward the limit shear strength; a similar consideration applies to PS loading conditions, since even in this case the normal confinement is statically determined.
However, computational (FE) models contain kinematic constraints arising from certain symmetries (consider plane strain or one-dimensional schemes) (Prevost, 1989; Borja et al., 1999; di Prisco et al., 2012). In addition, for SSI problems, where soil interacts with a (stiff) structural foundations and wall, the soil volume change plays an important role. The presence of kinematic constraints implies that some stress components are to be derived through compatibility conditions (e.g. prevented lateral expansion). That means that the local mean confinement is directly affected by the tendency of the material to dilate or contract. In particular, dilative frictional materials will increase the limit shear stress (with respect to unconfined conditions), while compactive frictional materials will decrease the limit shear stress. Further, not only the limit shear stress, but also the whole pre-failure response depends on the plastic flow rule whenever kinematic constraints are imposed (di Prisco and Pisanò, 2011; di Prisco et al., 2012).

The above considerations suggest that both experimental and numerical results are certainly affected by the kinematics of the system, even though this effect is not easy to be a priori quantified in terms of, for instance, $G/G_{\text{max}}$ and damping curves. The kinematic conditions of an infinite soil layer during 1D shear wave propagation are experimentally approximated through the well known “simple shear (SS) apparatus” (Wood, 2004), in which the soil specimen is cyclically sheared with no lateral expansion allowed. In order to assess how the kinematic confinement influences the cyclic response, stiffness degradation and damping curves are hereafter simulated under SS conditions by varying the volumetric
response of the soil; in particular, three different calibrations of the plastic flow rule (104.446) are considered, namely (i) isochoric \( k_d = \xi = 0 \), (ii) compactive \( k_d = M, \xi = 1 \) and (iii) dilative \( k_d = 0.4, \xi = 1 \).

The results reported in Figure 104.24 provide an insight into the possible effect of the volumetric response in combination with constrained loading conditions. In the isochoric case, the PS and the SS curves perfectly match (compare e.g. with the \( p_0 = 100 \) kPa curves in Figure 104.20), as, with no plastic expansion (or contraction), the lateral constraints do not affect the mean pressure during the shear loading; conversely, non-negligible SS-PS differences arise when dilative or contractive materials are considered. As is evident in Figure 104.24, the discrepancy between isochoric and non-isochoric curves becomes evident at medium/high cyclic strains, i.e. at the onset of significant plastifications. Indeed, while the mechanical response is barely inelastic, the deviatoric and the volumetric responses are practically decoupled, so that no variation of the normal confinement takes place.

Apparently, the quantitative significance of the above effects is strictly related to the actual dilational properties and confinement conditions. It can be in general concluded that the cyclic properties are expected to vary depending on specific loading conditions (triaxial, biaxial, simple shear, torsional shear, etc), so that, when numerical models are calibrated, this aspect should be always explicitly considered.

Figure 104.24: \( G/G_{max} \) and damping curves simulated under SS conditions and different volumetric responses (\( p_0 = 100 \) kPa, \( T = 2\pi \) s, \( G_{max} = 4 \) MPa, \( \nu = 0.25 \), \( M = 1.2 \), \( k_d = [1.2, 0.4] \), \( \xi = [0,1] \), \( h = G_{max}/(15p_0) \), \( m = 1 \) )
104.6.14.6 Concluding remarks

An incremental 3D elastic-plastic constitutive model was developed to reproduce the mechanical response of soils under cyclic/dynamic loading. The model is based on an effective-stress formulation with two parallel dissipative mechanisms, purely frictional (elastic-plastic) and viscous.

As far as the frictional mechanism is concerned, a bounding surface formulation with vanishing elastic region was adopted, extending to the case of pressure-sensitive non-associative soils the previous cohesive model by Borja and Amies (1994) for total-stress analysis. Notable features of the frictional model are: (i) the vanishing yield locus implies an elastic-plastic response at any load levels, as is observed in real experiments; (ii) a minimum number of physically meaningful parameters, which can be easily calibrated on the basis of a few experimental data; (iii) excellent performance and flexibility in reproducing in the elasto-plastic framework the standard stiffness degradation and damping curves. With reference to these latter, the parallel viscous mechanism – easy to be introduced in FE computations – was shown to provide an additional degree of freedom to improve the simulation of the cyclic energy dissipation, as long as the viscous parameter is properly calibrated. As a matter of fact, the viscous mechanism, used here, does physically exist in the form of viscous interaction between the soil solid skeleton and the pore fluid(s), and needs to be taken into account (as for example done here).

Future work will concern the investigation of the model performance in dynamic problems with pronounced hydro-mechanical coupling (cyclic mobility and liquefaction), as well as the comparison of the present model and traditional equivalent-linear approaches in seismic site response and SSI analysis. Further research is also needed to evaluate the accuracy of the model under non-symmetric loading conditions, these being particularly important in seismic slope stability applications.

104.6.14.7 Derivations of Various Equations

Derivation for Equation (104.449) The vanishing size of the yield locus implies:

\[
\lim_{k \to 0} s_{ij} = p\alpha_{ij} \Rightarrow ds_{ij} = d\alpha_{ij}p + \alpha_{ij}dp = \|d\alpha_{ij}\|n_{ij}^{dev}p + \alpha_{ij}dp
\]

and, after substituting the Prager translation rule (104.448):

\[
n_{ij}^{dev} = \frac{ds_{ij} - \alpha_{ij}dp}{\|d\alpha_{ij}\|p} = \frac{ds_{ij} - \alpha_{ij}dp}{\|ds_{ij} - \alpha_{ij}dp\|}
\]

In this last equality the property \(\|n_{ij}^{dev}\| = 1\) has been exploited.

Derivations for Equations (104.452) – (104.453) The consistency condition:

\[
df_y = 0 \iff \frac{\partial f}{\partial \sigma_{ij}}d\sigma_{ij} + \frac{\partial f}{\partial \alpha_{ij}}d\alpha_{ij} = 0
\]
results in the following equality chain
\[
\frac{\partial f}{\partial \sigma_{ij}} d\sigma_{ij} = 3p (s_{ij} - p\alpha_{ij}) \|d\alpha_{ij}\| n_{ij}^{\text{dev}} = 3p (s_{ij} - p\alpha_{ij}) \|d\alpha_{ij}\| \frac{3(s_{ij} - p\alpha_{ij})}{N^{\text{dev}}} = 0
\]  
(104.474)

leading to:
\[
\|d\alpha_{ij}\| = \frac{\partial f}{\partial \sigma_{ij}} d\sigma_{ij} pN^{\text{dev}}
\]  
(104.475)

where \(N^{\text{dev}} = \| (\partial f_y / \partial \sigma_{ij})^{\text{dev}} \| = 3\|s_{ij} - p\alpha_{ij}\|\). The above equation can be further simplified for the case of radial loading paths in the deviatoric plane, characterized by \(d\sigma_{ij} = ds_{ij}\) and coaxiality between the current stress state and its increment:
\[
\|d\alpha_{ij}\| = \frac{\partial f}{\partial \sigma_{ij}} ds_{ij} pN^{\text{dev}} = \frac{\partial f}{\partial \sigma_{ij}} ds_{ij} p\|s_{ij}\|
\]  
(104.476)

The final relationship can be re-expressed in terms of standards invariants \(q = \sqrt{3/2}\|s_{ij}\|\) and \(p\):
\[
\|d\alpha_{ij}\| = \sqrt{\frac{2}{3}} \frac{dq}{p}
\]  
(104.477)

**Derivation for Equation (104.470)** Under PS loading conditions (constant mean pressure), Equation (104.458) can be reduced to a simpler scalar form:
\[
\beta = \frac{\tau_{\lim} - \tau}{\tau - \tau_0}
\]  
(104.478)

where \(\tau_{\lim} = M\rho_0/\sqrt{3}\). By exploiting the previous definitions of deviatoric stress and strain invariants, the elastic-plastic response can be expressed as:
\[
d\epsilon = \frac{dq}{3G_{\text{max}}} + \left(\frac{\tau - \tau_0}{\tau_{\lim} - \tau}\right)^m \frac{dq}{\rho_0}
\]  
(104.479)

and specialized to the case of PS loading:
\[
\frac{d\gamma}{\sqrt{3}} = \frac{\sqrt{3}d\tau}{3G_{\text{max}}} + \left(\frac{\tau - \tau_0}{\tau_{\lim} - \tau}\right)^m \frac{\sqrt{3}d\tau}{\rho_0}
\]  
(104.480)

Integration over a strain interval between two stress reversals \((\gamma \in [-\gamma_{\text{max}}; \gamma_{\text{max}}])\) yields:
\[
2\gamma_{\text{max}} = \frac{2\tau}{G_{\text{max}}} + \frac{3}{\rho_0 \gamma_{\text{max}}} \int_{-\tau}^{\tau} \left(\frac{\tau'}{\tau_{\lim} - \tau'}\right)^m d\tau'
\]  
(104.481)

where \(\tau_0 = -\tau\) has been set. Straightforward variable changes lead to:
\[
1 = \frac{G}{G_{\text{max}}} + \frac{3}{2\rho_0 \gamma_{\text{max}}} \int_0^{2\gamma_{\text{max}} \gamma_{\text{max}}} \left(\frac{\tau''}{\tau_{\lim} - \tau'' + G_{\text{max}}}\right)^m d\tau''
\]  
(104.482)
1 = \frac{G_{\text{max}}}{G_{\text{max}}} \left[ 1 + \frac{6G_{\text{max}}}{hp_0\gamma_{\text{max}}} \int_0^{\gamma_{\text{max}}} \left( \frac{\gamma}{\gamma_{\text{lim}}/G - 2\gamma + \gamma_{\text{max}}} \right)^m d\gamma \right] \tag{104.483}

It is worth highlighting that two approximations are implicitly contained in Equation (104.483): (i) the integration over the first loading cycle does not exactly reproduce the stabilized cyclic response (because of the aforementioned Masing effect); (ii) a symmetric loading cycle in terms of shear strain does not in general ensure the symmetry of the corresponding shear stress range (as it is assumed in Equation (104.481)). However, such approximations do not prevent reasonable values for the hardening parameters $h$ and $m$ to be obtained.

### 104.6.15 Accelerated Constitutive Models

One of the concerns of elastic-plastic analysis is the computational effort. In this section developed are closed form elastic-plastic stress increments and elastic-plastic tangent stiffness matrices for commonly used elastic-plastic models. This is achieved by multiplying, in closed form, explicit, forward Euler elastic-plastic stress incremental solution and explicit, tangent elastic-plastic stiffness tensor, as developed in section 104.3 on page 192.

Relation between stress increment $d\sigma_{ij}$ and strain increments $d\epsilon_{kl}$ can be written for linear isotropic elasticity as:

$$d\sigma_{ij} = E_{ijkl}d\epsilon_{kl}$$ \tag{104.484}

where $E_{ijkl}$ is the fourth order elastic stiffness tensor. The elastic stiffness tensor features both minor symmetry $E_{ijkl} = E_{jikl} = E_{ijlk}$ and major symmetry $E_{ijkl} = E_{klij}$ (Jeremić and Sture (1997)). The elastic stiffness tensor for isotropic material can be written as:

$$E_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})$$ \tag{104.485}

where $\lambda$ and $\mu$ are the Lamé constants and $E$ and $\nu$ are Young’s Modulus and Poisson’s ratio, respectively:

$$\lambda = \frac{\nu E}{(1 + \nu)(1 - 2\nu)} \quad ; \quad \mu = \frac{E}{2(1 + \nu)}$$ \tag{104.486}

The relation between the stress tensor increment $d\sigma_{ij}$ and strain tensor increment $d\epsilon_{kl}$ is:

$$d\epsilon_{kl} = D_{klpq}d\sigma_{pq}$$ \tag{104.487}

where $D_{klpq}$ is the elastic compliance fourth order tensor, defined as:

$$D_{klpq} = -\frac{\lambda}{2\mu (3\lambda + 2\mu)} \delta_{kl}\delta_{pq} + \frac{1}{4\mu} (\delta_{kp}\delta_{lq} + \delta_{kq}\delta_{lp})$$ \tag{104.488}
104.6.15.1 Elasto-plasticity

Using developments from section 104.3 on page 192, one can write

\[ d\sigma_{ij} = E_{ijkl} d\epsilon_{kl} \]  \hspace{1cm} (104.489)

\[ d\epsilon_{ij}^p = d\lambda \frac{\partial Q}{\partial \sigma_{ij}} = d\lambda m_{ij}(\sigma_{ij}, q) \]  \hspace{1cm} (104.490)

\[ d\epsilon_{ij} = d\epsilon_{ij}^e + d\epsilon_{ij}^p \]  \hspace{1cm} (104.491)

\[ dq^* = d\lambda h(\tau_{ij}, q) \]  \hspace{1cm} (104.492)

where, \( \epsilon_{ij}, \epsilon_{ij}^e, \) and \( \epsilon_{ij}^p \) are total, elastic, and plastic strain tensors respectively, \( \sigma_{ij} \) is stress tensor, and \( q^* \) represents internal variables. Moreover, \( m_{ij} \) is the plastic flow direction, \( h \) is the plastic moduli, and \( d\lambda \) is a plastic consistency parameter that is to be determined. Equation (104.489) is the Hooke’s law that relates stress to elastic strain using the stiffness tensor \( E_{ijkl} \). Equation (104.490) shows relations of plastic strains to the associated or non-associated plastic flow rule. Equation (104.491) represents the additive decomposition of strain increment into elastic strain increment and plastic strain increments, that is applicable for small deformation analysis. Equation (104.492) represents evolution law for internal variables.

104.6.15.2 Elastic-Plastic Constitutive Models

For each elastic-plastic constitutive model, there are four components to be specified. Those components are (a) elasticity relation, (b) yield function, (c) plastic flow function, and (d) hardening and/or softening laws.

Once those four components are chosen, specified, elastic-plastic stiffness tensor, developed earlier in section 104.3 on page 192, can be developed for each model, by analytically multiplying functions and tensor equations for each component. In following sections this is done for von Mises, Drucker-Prager, and Cam-Clay material models.

The tangent elastic-plastic tensor is written as

\[ E_{p^el-pl}^{p^mm} = E_{pqmn} - \frac{E_{pqkl}m_{kl}m_{ij}E_{ijmn}}{m_{ot}E_{otrs}m_{rs} - \xi^* h^*} \]  \hspace{1cm} (104.493)
For perfectly plastic materials $\xi^*$ and $h^*$ would be zero while for the cases with evolution laws, the appropriate evolution laws should be used in derivation of tangent stiffness $E_{p,q,m,n}^{el-pl}$. Elastic modulus tensor is written as:

$$E_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})$$  \hspace{1cm} (104.494)

where $\lambda = \nu E/(1 + \nu)(1 - 2\nu)$, $\mu = E/(1 + \nu)$.

104.6.15.3 von Mises Model

von Mises yield criteria can be written as

$$f = [(s_{ij} - \alpha_{ij})(s_{ij} - \alpha_{ij})]^{0.5} - \sqrt{\frac{3}{2}}k = 0$$  \hspace{1cm} (104.495)

where $k$ is the scalar internal variable where initial value is related to the uniaxial tension strength, $\alpha_{ij}$ is the tensorial internal variable, the so called back stress that controls translational kinematic hardening.

The stress derivative of the yield function is

$$n_{ij} = \frac{\partial f}{\partial \sigma_{ij}} = \frac{1}{\sqrt{(s_{ij} - \alpha_{ij})(s_{ij} - \alpha_{ij})}}(s_{ij} - \alpha_{ij})$$  \hspace{1cm} (104.496)

In case of associated plastic flow rule where the plastic potential function is considered to be the same of yield function, stress derivative of plastic flow function is:

$$m_{ij} = \frac{\partial g}{\partial \sigma_{ij}} = \frac{1}{\sqrt{(s_{ij} - \alpha_{ij})(s_{ij} - \alpha_{ij})}}(s_{ij} - \alpha_{ij})$$  \hspace{1cm} (104.497)

In order to find the closed form equation for elastic-plastic modulus for case of perfectly-plastic, Equation (104.495) is used for both yield surface and plastic potential and no evolution law is considered. Relevant stress derivatives are:

$$n_{ij} = \frac{1}{\sqrt{s_{ij} s_{ij}}} s_{ij} ; \quad m_{ij} = \frac{1}{\sqrt{s_{ij} s_{ij}}} s_{ij}$$  \hspace{1cm} (104.498)

Base on obtained stress derivatives, plastic consistency parameter is derived as

$$d\lambda = \frac{n_{ij} E_{ijkl} d\epsilon_{pq} m_{kl}}{n_{ij} E_{ijkl} m_{kl}} = \frac{n_{ij} d\sigma_{ij}}{2\mu}$$  \hspace{1cm} (104.499)
The elastic-plastic tangent tensor is then

\[ E_{pqmn}^{pl} = \frac{E_{pqkl} n_{ab} E_{abcd} m_{cd}}{n_{ab} E_{abcd} m_{cd}} = 2\mu n_{pq} n_{mn} \]

\[ = 2\mu \left( \frac{1}{\sqrt{s_{kl} s_{kl}}} s_{pq} \frac{1}{\sqrt{s_{ij} s_{ij}}} s_{mn} \right) = 2\mu \left( \frac{1}{s_{kl} s_{kl}} s_{pq} s_{mn} \right) \]

(104.500)

In case isotropic hardening, \( \xi \) and \( h \) are not zero anymore, since \( k \) is scalar internal variable that is updated in each increment. Hardening/softening functions, scalar functions in this case. \( \xi \) and \( h \) can be calculated as

\[ \xi = -\sqrt{\frac{2}{3}} \]

(104.501)

\[ h = \frac{2 \mu}{\sqrt{3} k} m_{ij} m_{ij} \]

(104.502)

Therefore, plastic parameter \( (d\lambda) \) and tangent elastic-plastic tensor \( (E_{pqmn}^{pl}) \) can be written as:

\[ d\lambda = \frac{n_{ij} E_{ijpq} d\sigma_{pq}}{n_{ij} E_{ijkl} m_{kl} - \xi h} = \frac{n_{ij} d\sigma_{ij}}{2\mu + \frac{2}{3} k} \]

(104.503)

\[ E_{pqmn}^{pl} = \frac{E_{pqkl} n_{ab} E_{abcd} m_{cd}}{n_{ab} E_{abcd} m_{cd} - \xi h} = \frac{4\mu^2 n_{pq} n_{mn}}{2\mu + \frac{2}{3} k} \]

\[ = \frac{4\mu^2}{\sqrt{s_{kl} s_{kl}}} s_{pq} \frac{1}{\sqrt{s_{ij} s_{ij}}} s_{mn} = \frac{4\mu^2}{s_{kl} s_{kl}} s_{pq} s_{mn} \]

(104.504)

The value of \( k \) should be updated at each step using the following equation:

\[ k_{updated} = k + d k = k + h d\lambda = k + \frac{\sqrt{2}}{3} kd\lambda = k \left( 1 + \sqrt{\frac{2}{3}} d\lambda \right) \]

(104.505)

For case the of von Mises with kinematic hardening, \( \alpha_{ij} \) is the tensorial internal variable to be updated at each step of analysis, while tensor functions \( \xi_{ab} \) and \( h_{ab} \) can be calculated from the following equations using Armstrong-Frederick saturation-type kinematic hardening rule (Armstrong and Frederick (1966); Lemaitre and Chaboche (1990)):
\[ \xi_{ab} = \frac{\partial F}{\partial \alpha_{ab}} = \frac{1}{\sqrt{(s_{ab} - \alpha_{ab})(s_{ab} - \alpha_{ab})}}(s_{ab} - \alpha_{ab}) \] (104.506)

Recall that \( dq_{ij} = d\alpha_{ij} = d\lambda h_{ij} \), and \( d\epsilon_{ij}^{pl} = d\lambda m_{ij} \) and that the Armstrong-Frederic kinematic hardening rule for the internal variable \( \alpha_{ij} \) is given as:

\[ d\alpha_{ij} = \frac{2}{3} h_{ij} \left( d\epsilon_{ij}^{p} \right)^{dev} - c_r \alpha_{ij} \sqrt{\frac{2}{3} \left( d\epsilon_{st}^{p} \right)^{dev} \left( d\epsilon_{st}^{p} \right)^{dev}} \] (104.507)

so that

\[ h_{st} = \frac{2}{3} h_{ij} m_{st} - \alpha_{st} c_r \sqrt{\frac{2}{3}} \] (104.508)

since \( \sqrt{m_{ij} m_{ij}} = 1 \).

Plastic consistency parameter \((d\lambda)\) and tangent elastic-plastic tensor \(E_{pqmn}^{pl}\) can be expressed as:

\[ d\lambda = \frac{n_{ij} E_{ijpq} d\epsilon_{pq}}{n_{ij} E_{ijkl} m_{kl} - \xi h} = \frac{n_{ij} d\sigma_{ij}}{2\mu - \xi_{ab} h_{ab}} \] (104.509)

\[ E_{pqmn}^{pl} = \frac{E_{pqkl} n_{kl} m_{ij} E_{ijmn}}{n_{ab} E_{abcd} m_{cd} - \xi h} = \frac{4\mu^2 n_{pq} n_{mn}}{2\mu - \xi_{ij} h_{ij}} \] (104.510)

Full, developed form of tangent elastic-plastic stiffness can be written as
\[ E_{pqmn}^p = E_{pqkl} \left( \frac{1}{\sqrt{(s_{ab} - \alpha_{ab})(s_{ab} - \alpha_{ab})}} (s_{kl} - \alpha_{kl}) \right) \left( \frac{1}{\sqrt{(s_{ab} - \alpha_{ab})(s_{ij} - \alpha_{ij})}} (s_{ij} - \alpha_{ij}) \right) E_{ijmn} \]

\[ \left( \frac{1}{\sqrt{(s_{mn} - \alpha_{mn})(s_{mn} - \alpha_{mn})}} (s_{ab} - \alpha_{ab}) \right) E_{abcd} \left( \frac{1}{\sqrt{(s_{mn} - \alpha_{mn})(s_{mn} - \alpha_{mn})}} (s_{cd} - \alpha_{cd}) \right) - \left( \frac{1}{\sqrt{(s_{ab} - \alpha_{ab})(s_{ab} - \alpha_{ab})}} (s_{rs} - \alpha_{rs}) \right) \left( \frac{2}{3} h_{ab} h_{cd} - \frac{2}{3} \alpha_{rs} c_r \right) \]

\[ (104.511) \]
Above equation can be simplified, written in a shorter form by using a substitution

\[ \mathcal{A} = \frac{1}{\sqrt{(s_{ab} - \alpha_{ab})(s_{cd} - \alpha_{cd})}} \]  

(104.512)

it can be written as:

\[
E_{pqmn}^{pl} = \frac{E_{pqkl} \left( \mathcal{A} (s_{kl} - \alpha_{kl}) \right) \left( \mathcal{A} (s_{ij} - \alpha_{ij}) \right) E_{ijmn}}{(s_{ab} - \alpha_{ab}) E_{abcd} \left( \mathcal{A} (s_{cd} - \alpha_{cd}) \right) - \left( -\mathcal{A} \right) \left( \frac{2}{3} h a \left( \mathcal{A} (s_{cd} - \alpha_{cd}) \right) - \sqrt{\frac{2}{3} \alpha_{cd} c_r} \right) (s_{cd} - \alpha_{cd})}
\]  

(104.513)

Finally the full elastic-plastic tangent stiffness tensor can be written as:

\[
E_{pqmn}^{el-pl} = E_{pqmn}^{el} - E_{pqmn}^{pl} = \frac{E_{pqkl} \left( \mathcal{A} (s_{kl} - \alpha_{kl}) \right) \left( \mathcal{A} (s_{ij} - \alpha_{ij}) \right) E_{ijmn}}{(s_{ab} - \alpha_{ab}) E_{abcd} \left( \mathcal{A} (s_{cd} - \alpha_{cd}) \right) - \left( -\mathcal{A} \right) \left( \frac{2}{3} h a \left( \mathcal{A} (s_{cd} - \alpha_{cd}) \right) - \sqrt{\frac{2}{3} \alpha_{cd} c_r} \right) (s_{cd} - \alpha_{cd})}
\]  

(104.514)

104.6.15.4 Nonlinear Elastic Model in 1D based on Armstrong Frederick Equation

\[
d\sigma_{ij} = d\alpha_{ij} = \frac{2}{3} h a \left( d\epsilon_{ij}^p \right)^{dev} - c_r \sigma_{ij} \sqrt{\frac{2}{3} (d\epsilon_{st}^p)^{dev} (d\epsilon_{st}^p)^{dev}}
\]  

(104.515)

104.6.15.5 Drucker-Prager Model

Drucker-Prager model yield surface, including rotational kinematic hardening can be written as

\[
f = \alpha I_1 + \frac{1}{2} (s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})^{\frac{1}{2}} - \beta = 0
\]  

(104.516)

where \( \alpha \) and \( \beta \) are material constants.

By coinciding Drucker-Prager cone with the outer apexes of the Mohr-Coulomb hexagon locus, the constants for compressive cone of Drucker-Prager can be evaluated as shown in Equation (104.517) by knowing the soil strength parameters of cohesion (c) and friction angle (\( \phi \)):

\[
\alpha = \frac{2 \sin \phi}{\sqrt{3(3 - \sin \phi)}} , \quad \beta = \frac{6 \cos \phi}{\sqrt{3(3 - \sin \phi)}} c
\]  

(104.517)
Drucker-Prager yield function for cohesionless sands \((k = 0)\) can be obtained as:

\[
f = \alpha I_1 + \sqrt{J_2} = 0 \quad \text{(104.518)}
\]

or in \(p - q\) space:

\[
f = q - Mp = 0 \quad \text{(104.519)}
\]

which then \(M\) can be obtained as:

\[
M = \frac{6 \sin \phi}{3 - \sin \phi} \quad \text{(104.520)}
\]

By considering kinematic hardening, Equation (104.519) can be expressed as:

\[
f = \left[(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij}) \right]^{0.5} - \sqrt{\frac{2}{3}} kp = 0 \quad \text{(104.521)}
\]

which then stress derivative of yield function and plastic potential function in case of associated plasticity can be defined as:

\[
n_{ij} = m_{ij} = \frac{\partial f}{\partial \sigma_{ij}} = \frac{\partial g}{\partial \sigma_{ij}}
\]

\[
= \left[ (s_{ij} - p\alpha_{ij}) + \frac{1}{3} \alpha_{pq}\delta_{ij} (s_{pq} - p\alpha_{pq}) \right] \left[(s_{rs} - p\alpha_{rs})(s_{rs} - p\alpha_{rs}) \right]^{-0.5}
\]

\[
+ \sqrt{\frac{2}{27}} k\delta_{ij} \quad \text{(104.522)}
\]

To find the closed form equation of elastic-plastic modulus for case of perfectly-plastic, Equation (104.521) is used for associated plasticity rule with no hardening. By these assumptions stress derivative is calculated as:

\[
n_{ij} = m_{ij} = \frac{1}{\sqrt{s_{ij}s_{ij}}} s_{ij} + \sqrt{\frac{2}{27}} k\delta_{ij} \quad \text{(104.523)}
\]

Splitting plastic modulus to different parts:

\[
Hq_{ij} = E_{ijkl} m_{kl} = \{\lambda\delta_{ij}\delta_{kk} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\}\left\{\frac{1}{\sqrt{s_{mn}s_{mn}}} s_{kl} + \sqrt{\frac{2}{27}} k\delta_{kl}\right\}
\]

\[
= 2\mu \frac{1}{\sqrt{s_{mn}s_{mn}}} s_{ij} + \delta_{ij} \left\{\frac{1}{\sqrt{s_{mn}s_{mn}}} \lambda (s_{pq}\delta_{pq}) + \sqrt{\frac{2}{27}} k(3\lambda + 2\mu)\right\}
\]

\[
= 2\mu \frac{1}{\sqrt{s_{mn}s_{mn}}} s_{ij} + B\delta_{ij} \quad \text{(104.524)}
\]
Assuming:

\[ B = \frac{1}{\sqrt{s_{mn}s_{mn}}} \lambda (s_{pq}\delta_{pq}) + \sqrt{\frac{2}{27}} k(3\lambda + 2\mu) \]  

(104.525)

Plastic parameter can be written as:

\[ d\lambda = \frac{n_{ij}E_{ijpq}d\epsilon_{pq}}{n_{ij}E_{ijpq}m_{kl}} = \frac{\lambda_{nominator}}{\lambda_{denominator}} \]  

(104.526)

\[ \lambda_{nominator} = n_{ij}E_{ijpq}d\epsilon_{pq} = n_{ij}d\sigma_{ij} \]  

(104.527)

\[ \lambda_{denominator} = n_{ij}E_{ijpq}m_{kl} = n_{ij}H_{ij} \]

\[ = \left\{ \frac{1}{\sqrt{s_{ij}s_{ij}}} s_{ij} + \sqrt{\frac{2}{27}} k\delta_{ij} \right\} \left\{ 2\mu \frac{1}{\sqrt{s_{mn}s_{mn}}} s_{ij} + B\delta_{ij} \right\} \]

\[ = 2\mu + \frac{1}{\sqrt{s_{mn}s_{mn}}} (B + 2\mu\sqrt{\frac{2}{27}} k) (s_{ij}\delta_{ij}) + 3B\sqrt{\frac{2}{27}} k \]  

(104.528)

Finally plastic tensor can be expressed as:

\[ E_{pl}^{pqmn} = \frac{E_{pqkm}n_{kl}E_{ijmn}}{n_{ab}E_{abcd}m_{cd}} = \frac{H_{pq}H_{mn}}{\lambda_{denominator}} \]  

(104.529)

where:

\[ H_{pq}H_{mn} = \left\{ -\frac{2\mu}{\sqrt{s_{ij}s_{ij}}} s_{pq} + B\delta_{pq} \right\} \left\{ -\frac{2\mu}{\sqrt{s_{ij}s_{ij}}} s_{mn} + B\delta_{mn} \right\} \]

\[ = 4\mu^2 \frac{1}{s_{ij}s_{ij}} s_{pq}s_{mn} + 4\mu \frac{1}{\sqrt{s_{ij}s_{ij}}} B s_{pq}\delta_{mn} + B^2 \delta_{pq}\delta_{mn} \]  

(104.530)

When isotropic hardening is considered for Drucker-Prager model, \( \xi \) and \( h \) are not zero. These parameters can be obtained using following equations:

\[ \xi = -p\sqrt{\frac{2}{3}} \]  

(104.531)
\[ h = \sqrt{\frac{2}{3} m_{ij} m_{ij} k} = \sqrt{\frac{2}{3} (1 + \frac{2}{9} k^2) k} \]  

Plastic parameter \((d\lambda)\) and plastic modulus \((E_{pqmn}^{pl})\) can be written as:

\[ \lambda_{\text{denominator}} = n_{ij} E_{ijpq} m_{kl} - \xi h = n_{ij} H q_{ij} - \xi h \]  

\[ E_{pqmn}^{pl} = \frac{E_{pqkt} m_{ijkl} m_{kl} - \xi h}{n_{ab} E_{abcd} m_{cd}} = \frac{H q_{pq} H q_{mn}}{\lambda_{\text{denominator}}} \]  

\(k\) is the scalar internal variable to be updated at each step of analysis using following equation:

\[ k_{\text{updated}} = k + dk = k + h d\lambda = k + \sqrt{\frac{2}{3} (1 + \frac{2}{9} k^2) k d\lambda} = k(1 + \sqrt{\frac{2}{3} (1 + \frac{2}{9} k^2) d\lambda}) \]  

By considering kinematic hardening for Drucker-Prager material model, tensorial internal variable \((\alpha_{ij})\) is introduced and has to be updated at each step of analysis. The stress derivatives of yield function and plastic potential function can be written as:

\[ n_{ij} = m_{ij} = \frac{1}{\sqrt{(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})}} (s_{ij} - p\alpha_{ij}) + \sqrt{\frac{2}{27}} k \delta_{ij} \]  

\[ \xi_{ij} = -\frac{p}{\sqrt{(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})}} (s_{ij} - p\alpha_{ij}) \]  

\[ h_{ij} = \frac{2}{3} h_{a} m_{ij} - \alpha_{ij} c_r \sqrt{\frac{2}{3} m_{ij} m_{ij}} \]  

Plastic parameter \((d\lambda)\) and plastic modulus \((E_{pqmn}^{pl})\) are obtained from following equations:

\[ d\lambda = \frac{n_{ij} E_{ijpq} d\epsilon_{pq}}{n_{ij} E_{ijkl} m_{kl} - \xi_{ij} h_{ij}} = \frac{n_{ij} d\sigma_{ij}}{2\mu - \xi_{ij} h_{ij}} \]
\[ E_{pqmn}^{pl} = \frac{E_{pqkl}^{nk}m_{ij}E_{ijmn}}{n_{ab}E_{abcd}m_{cd} - \xi_{ij} h_{ij}} \]  

(104.540)

Tensorial internal variable \((\alpha_{ij})\) can be updated using following equation:

\[
\alpha_{ij}^{updated} = \alpha_{ij} + d\alpha_{ij} = \alpha_{ij} + h_{ij}d\lambda = \alpha_{ij} + \left(\frac{2}{3}h_{a}m_{ij} - \alpha_{ij}c_{r}\sqrt{\frac{2}{3}m_{ij}m_{ij}}\right)d\lambda
\]  

(104.541)

### 104.6.15.6 Modified Cam-Clay Model

The critical state line for Cam-Clay can be written as

\[
e_{c} = e_{c,r} - \lambda_{c} \ln p_{c}
\]  

(104.542)

where \(e_{c}\) is the critical void ratio at critical mean stress \((p_{c})\), \(e_{c,r}\) is the reference critical void ratio, and \(\lambda_{c}\) is the normal consolidation slope. In general it is assumed that the normal consolidation line (NCL) is parallel to CSL, which is defined as:

\[
e = e_{\lambda} - \lambda \ln p
\]  

(104.543)

where \(e_{\lambda}\) is the intercept on the NRL at \(p = 1\). \(\lambda\) is the normal consolidation slope or the elasto-plastic slope of \(e - \ln p\) relation. The same relation is used for unloading-reloading line (URL) with different slope as:

\[
e = e_{\kappa} - \kappa \ln p
\]  

(104.544)

where \(e_{\kappa}\) is the intercept on the URL at \(p = 1\). The yield function for Cam-Clay model can be defined as

\[
f = q^{2} - M^{2}[p(p_{0} - p)] = 0
\]  

(104.545)

where \(M\) is the critical state stress ratio in \(q - p\) space and \(p_{0}\) is the initial internal scalar variable which will be changed by change in plastic volumetric strain.

Cam-Clay model is one of the associated flow rule models which means the same function is used for both yield and plastic potential surfaces \((f = g)\). The plastic flow of the Cam-Clay model is associated
with its yield function, in other words, the plastic flow is defined by the potential function \( g \), which is assumed the same as the yield function \( f \).

The evolution law of for Cam-Clay model is a scalar one which can be expressed by:

\[
\dot{p}_0 = \frac{(1 + e) p_0}{\lambda - \kappa} \epsilon_v
\]

(104.546)

where \( e \) is the void ratio, \( \lambda \) is the normal consolidation slope or the elasto-plastic slope of \( e - \ln p \) relation, and \( \kappa \) is the slope of unloading-reloading line. This equation proves that the change of \( p_0 \) is controlled by change of plastic volumetric strain. By considering the yield function expressed in Equation (104.545) and considering the associated flow rule, stress derivatives can be evaluated as:

\[
\sigma_{ij} = m_{ij} = 3s_{ij} + \frac{1}{3} M^2 (p_0 - 2p) \delta_{ij}
\]

(104.547)

Elastic modulus in terms of shear modulus \( G \) and bulk modulus \( K \) can be expressed as:

\[
E_{ijkl} = (K - \frac{2}{3} G) \delta_{ij} \delta_{kl} + G (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})
\]

(104.548)

\[
H_{ij} = E_{ijkl} m_{kl}
\]

\[
= \{(K - \frac{2}{3} G) \delta_{ij} \delta_{kl} + G (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})\}\{3s_{ij} + \frac{1}{3} M^2 (p_0 - 2p) \delta_{ij}\}
\]

\[
= 6Gs_{ij} + \{(K - \frac{2}{3} G) M^2 (p_0 - 2p) + \frac{2}{3} M^2 (p_0 - 2p) G\} \delta_{ij}
\]

(104.549)

Defining \( D \) as:

\[
D = (K - \frac{2}{3} G) M^2 (p_0 - 2p) + \frac{2}{3} M^2 (p_0 - 2p) G
\]

(104.550)

\( \xi \) and \( h \) in case of using Cam-Clay constitutive model can be expressed as following equations:

\[
\xi = M^2 p
\]

(104.551)

\[
h = (1 + e_0) p_0 \frac{d\epsilon_v}{\lambda - \kappa}
\]

(104.552)

where \( d\epsilon_v = M^2 (2p - p_0) \). Plastic parameter can be written as:
\[ d\lambda = \frac{n_{ij}E_{ijpq}d\epsilon_{pq}}{n_{ij}E_{ijpq}m_{kl} - \xi h} = \frac{\lambda_{\text{nominator}}}{\lambda_{\text{denominator}}} \] (104.553)

\[ \lambda_{\text{nominator}} = n_{ij}E_{ijpq}d\epsilon_{pq} = n_{ij}d\sigma_{ij} \] (104.554)

\[ \lambda_{\text{denominator}} = n_{ij}E_{ijpq}m_{kl} - \xi h = n_{ij}Hq_{ij} - \xi h \]
\[ = \{3s_{ij} + \frac{1}{3}M^2(p_0 - 2p)\delta_{ij}\}\{6G_{s_{ij}} + D\delta_{ij}\} - \xi h \]
\[ = 18G_{s_{ij}}s_{ij} + DM^2(p_0 - 2p) - \xi h \] (104.555)

Then the plastic tensor can be expressed as:

\[ E_{pqmn}^{pl} = \frac{E_{pqkl}m_{kl}m_{ij}E_{ijmn}}{n_{ab}E_{abcd}m_{cd} - \xi h} = \frac{H_{pq}Hq_{mn}}{\lambda_{\text{denominator}}} \] (104.556)

where:

\[ H_{pq}Hq_{mn} = \{6G_{s_{ij}} + D\delta_{ij}\}\{6G_{s_{ij}} + D\delta_{ij}\} \]
\[ = D^2\delta_{pq}\delta_{mn} + 6GD\delta_{pq}s_{mn} + 6GDs_{pq}\delta_{mn} + 36G^2s_{pq}s_{mn} \] (104.557)

\[ p_0 \] is the scalar internal variable to be updated at each step of analysis using the following equation:

\[ p_0^{\text{updated}} = p_0 + dp_0 = p_0 + h d\lambda \] (104.558)

104.6.15.7 Comparison of Computational Time of Accelerated Constitutive Models with NewTemplate3Dep

In order to compare the computational time of accelerated constitutive models with the ones available in NewTemplate3Dep, cyclic simulations are done in constitutive level using explicit integration method. Simulations are done for all the mentioned cases of von Mises, Drucker-Prager, and Cam-Clay constitutive models. The ratios of computational time of NewTemplate3Dep \( (t_N) \) to accelerated constitutive models \( (t_{\text{Acc}}) \) are summarized in Table (104.1). As it is observed, there are improvements in range of 2 to 3 times in computational time which can lead to reducing the computational time of soil-structure systems.
Table 104.1: Comparison of computational time of accelerated constitutive models with NewTem-plate3Dep

<table>
<thead>
<tr>
<th>Constitutive Model</th>
<th>$t_N/t_{Acc}$</th>
<th>Constitutive Model</th>
<th>$t_N/t_{Acc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>von Mises Perfectly Plastic</td>
<td>3.1</td>
<td>Drucker-Prager Perfectly Plastic</td>
<td>2.9</td>
</tr>
<tr>
<td>von Mises Isotropic Hardening</td>
<td>2.6</td>
<td>Drucker-Prager Isotropic Hardening</td>
<td>2.5</td>
</tr>
<tr>
<td>von Mises Kinematic Hardening</td>
<td>2.3</td>
<td>Drucker-Prager Kinematic Hardening</td>
<td>1.9</td>
</tr>
<tr>
<td>Cam-Clay</td>
<td>2.2</td>
<td>——</td>
<td>——</td>
</tr>
</tbody>
</table>

104.7 Elastic-Plastic Models for Contacts, Joints and Interfaces

This section is based on Sinha and Jeremić (2017).

104.7.1 Experimental Data

The response of the interface plays a very important role on the behavior of deep and shallow foundations, retaining walls, geo-membranes, submerged structures and soil-structure interaction. The load transfer mechanism from structure to the soil acts at the interface. The soil-structure interface comprise of a very thin small shearing band. The initial investigation by Yoshimi and Kishida (1981) indicated the thickness of the shear band as nine times the mean grain size diameter $D_{50}$. Tejchman and Wu (1995) conducted several tests on sand-steel interface and concluded that the thickness of the interface for rough interface is 30-40 $D_{50}$ and for smooth interface is 6-10 $D_{50}$. Dejong et al. (2006); DeJong and Westgate (2009) investigated the shear-zone thickness to be 5-10 times the mean particle diameter $D_{50}$. Martinez et al. (2015) conducted several axial and torsional shear experiments on sand-steel interface to understand the micro mechanics involved at interface. The micro-mechanical hypothesis proposed by Martinez et al. (2015) is shown in Figure 104.25. The thickness of the interface zone formed by the particle in axial shearing migrates along the interface whereas for torsional shearing it migrates away from the interface. Similar mechanical hypothesis was observed by Dejong et al. (2006) It was also observed that for torsional shear the shear band zone was 2-3 times larger than the purely axial shearing. It also depends upon the characteristics of the soil beneath and the surface structural material and its roughness.

Since 1960’s, researchers have been carrying out experiments to understand the interface behavior. The initial works have been contributed by Potyondy (1961); Brumund and Leonards (1973); Littleton...
Potyondy (1961) studied the effect of soil-moisture content, structural surface roughness, soil-composition and normal load intensity on the skin-friction of the soil-structure interface. Brumund and Leonards (1973) investigated the static and dynamic friction angle between sand steel interface. Littleton (1976) performed drained and undrained tests on clay-steel interface and found that the shear stress response was steeper than the usual clay-clay interface. Based on the normal confinement $\sigma_n$, an initial hardening was observed until the peak shear strength $\tau_p$ is reached. After that softening to residual $\tau_r$ was observed. Later Desai (1981) emphasized on the importance of modeling of interface behavior for real soil-structure interactions. He also pointed out the lack of existing experimental data which could be used to develop constitutive models defining the interface behavior. Yoshimi and Kishida (1981) used a ring torsion apparatus to find the friction angle between dry sand and steel surface over a wide variation of surface roughness and sand density. Uesugi and Kishida (1986a,b); Kishida and Uesugi (1987) carried a series of laboratory experiments between steel and air-dried sands using simple shear apparatus shown in Figure 104.26 and Figure 104.27. It was found that the interface behavior is highly influenced by the sand type and surface roughness $R_{\text{max}}$ while the effect of normal stress $\sigma_n$ and mean grain size $D_{50}$ are of poor significance. Thus, Kishida and Uesugi (1987) proposed a normalized roughness $R_{n}$ to evaluate the relative coefficient of friction $\mu_y$ of sand-steel interface as shown in Figure ??.

$$R_{\text{max}}(L = D_{50})/D_{50}$$

(104.559)

where, $R_{\text{max}}(L = D_{50})$ is the $R_{\text{max}}$ value of the steel surface with gauge length $L = D_{50}$.

Desai and Nagaraj (1988) performed a cyclic normal and shear tests on dry sand and concrete interface in translational shear box. Monotonic and cyclic normal loads along with cyclic shear loads

Figure 104.25: Hypothesis for a particle movement under (a) axial and (b) torsional loading (Martinez et al. (2015))
Figure 104.26: Section of friction test apparatus (Uesugi and Kishida (1986b)).

were applied. The shear behavior was modeled with a modified form of Ramberg-Osgood (R-O) model. Although Desai and Nagaraj (1988) did not show any experimental results, he idealized the normal interface behavior to be composed of (1) Virgin loading; (2) unloading; (3) reloading; 4 tensile condition; (5) partial debonding; and 6 rebonding as shown in Figure 104.29. Uesugi et al. (1989, 1990) studied the frictional behavior of sand-steel interface subject to repeated shearing under one-way or two-way loadings. It was found that under repeated loading conditions the coefficient of friction $\mu$ converged close to the residual coefficient of friction $\mu_r$ as could be observed in Figure ?? Boulon (1989) performed a lot of experiments on piles in sand. Direct simple shear tests were carried out to investigate the shear behavior between the granular soil and rough construction material. Based on the experimental results obtained later Boulon and Nova (1990) proposed a mathematical model and constitutive integration to model the interface behavior in finite element method (FEM). Aubry et al. (1990) proposed a dilatancy based cyclic elastic-plastic constitutive model for the interface. Cyclic loading functions with memory of last loading reversal was used to model subsequent loadings and unloading. The yield function was defined using simple Mohr Coulomb with additional parameter $F$ as a function of normal stress $\sigma_n$ and plastic compressibility $\beta$ to account for curvature and dilation of the yield surface.

Fakharian and Evgin (1995) developed a 3-D apparatus capable of performing direct and simple shear type testing of interfaces between soil and structure. The developed apparatus was subsequently used to perform numerous experiments Fakharian and Evgin (1996); Evgin and Fakharian (1997); Fakharian and Evgin (1997); Fakharian (1996); Fakharian et al. (2002) over sand-steel interface for different stress paths for different relative densities $D_r$ of sand. The 3-D apparatus made it possible to conduct 2-D
Figure 104.27: Measurement of tangential displacement (Uesugi and Kishida (1986b)).

Figure 104.28: Coefficient of friction at yield $\mu_y$ and normalized roughness (after Kishida and Uesugi (1987)).
shear test with constant normal stress \( \sigma_n \). Monotonic and cyclic test results are shown in Figure 104.31 and Figure 104.31. The experiment results in Figure 104.31 clearly shows a peak shear stress ratio \( (\tau/\sigma_n)_p \) and a residual stress ratio \( (\tau/\sigma_n)_r \). Initial hardening and then softening depends upon the relative density \( D_r \) of the sand was observed. A higher relative density \( D_r = 80\% \) sand shows dilation for lower confinement and thus a peak behavior whereas a low relative density \( D_r = 25\% \) sand shows no dilation. For cyclic shear tests, the loose soil \( D_r = 25\% \) shown in Figure 104.32(b) showed gain in strength due to densification resulting from particle breakage. While for soil with high relative density \( D_r = 80\% \) almost no gain in shear strength was observed during cyclic shearing. The 3-D tests performed showed that the shear stress \( \tau \) is almost isotropic for different shear stress paths.

Shahrour and Rezaie (1997) performed a series of monotonic and cyclic tests on Hostun Sand with rough and smooth surface with constant normal load condition. The results obtained were used to propose an elasto-plastic bounding surface based constitutive model for the interface behavior. The monotonic and cyclic test are shown in Figure 104.33 for rough and smooth interface surface. From Figure 104.33, it could be observed that for smooth interface, the shear stress \( \tau \) increases only upto the critical shear stress \( \tau_c \). Whereas for rough soil, the shear stress \( \tau \) hardens to a peak strength \( \tau_p \) and then softens to the critical shear strength \( \tau_c \). The behavior observed for rough and smooth interface is similar to dense and loose soil as observed in tests by Fakharian and Evgin (1996). The cyclic test shown in Figure 104.33(c) showed similar response as was observed by Fakharian and Evgin (1996).

Dejong et al. (2006) performed laboratory soil-structure investigation under constant normal stiffness using particle image velocimetry (PIV) method. Cyclic shearing was carried out to simulate and understand the the shear behavior at pile interface. Cyclic degradation exponential model was proposed of model the change in void ratio. Later Mortara et al. (2007) also performed cyclic shear tests on sand
Figure 104.30: (a) Monotonic and (b) cyclic response of Toyora sand with steel interface (Uesugi et al. (1989)) \((D_r = 90\%, \sigma_n = 98kPa, R_n = 150e^{-3})\).

steel interfaces. DeJong and Westgate (2009) quantified the soil-structure interface behavior to the shearing on the factor like relative density \(D_r\), particle angularity, particle hardness, surface roughness, normal stress and normal stiffness. Local as well as global load displacement response was recorded to understand the load-transfer mechanism.

104.7.2 Axial Contact, Joint, Interface

The contact/joint/interface behavior in the normal direction is modeled as penalty stiffness function as described in the Section 104.7.2.1. The penalty function can be chosen as linear with fixed stiffness also known as Hard Contact/Joint/Interface (Section 104.7.2.2), or it can be assumed to be a non-linear function with stiffness increasing exponentially with penetration. This type of normal behavior is called as Soft Contact/Joint/Interface (Section 104.7.2.3). Soft Contact/Joint/Interface represents more realistic soil-structure interface behavior. The soil becomes stiff as the penetration increases and gets relaxed upon unloading.

104.7.2.1 Penalty Method

At the interface of the soil-foundation system, an impenetrability constraint exists as shown in Equation ?? to ???. The contacting/interfacing bodies cannot penetrate into each other. The impenetrability leads to an inequality constraint, which requires special methods such as penalty method, Lagrange,
penalty method is a common approach used for solving constrained minimization (or maximization) problems involving inequalities as described in section ?? and Section ??

In this approach, a large penalty term is added to the minimizing functional to prevent the solution from escaping the constrained space.

Figure 104.34 shows a two contact/interface node pairs initially separated by a small distance of $g$ in the contact/interface normal direction. During pure contact/interface/joint, the two node penetrates against each other by $\Delta_n$. The instantaneous relative distance between the two contact/interface surfaces is $u$. Thus, if $u < g$ there is no contact/interface and normal force $N = 0$ otherwise there is contact/interface and a normal force $N$ will act.

In the penalty stiffness formulation, a small penetration $\Delta_n$ is allowed between the mass and the floor having stiffness $k_n$ such that during contact the normal force $N$ is defined as

$$N = k_n \Delta_n \quad \text{if} \quad \Delta_n \geq 0$$

(104.560)

where $k_n$ can be thought of the normal contact/interface stiffness and $\Delta_n$ is the relative displacement between the two contact/interface surfaces with respect to the initial gap $g$ in contact/interface normal direction. It is defined as the following

$$\Delta_n = u_n^r - g_n = u - g$$

(104.561)

where $u_n^r$ is the relative displacement in the contact/interface normal direction and $g_n$ is the initial gap in contact/interface normal direction. Theoretically, for rigid contact/interface case, the penalty stiffness $k_n$ is assumed to infinite resulting in $\Delta_n = 0$. However, for numerical reasons, infinity is not possible, and
Figure 104.32: Cyclic test results, shear stress versus shear displacement for $\sigma_n = 500\, kPa$: (a) rough surface, dense sand ($D_r = 80\%$); (b) rough surface, loose sand ($D_r = 25\%$) (Fakharian and Evgin (1996)).

thus $\Delta_n = 0$ is never enforced. This results in small penetration at contact/interface surfaces resulting in $\Delta_n < 0$ during contact. For penalty method, the term penetration is thus normally referred to $\Delta_n$ defining the two possible states as:

- **No Contact/Joint/Interface** (Penetration $\Delta_n > 0$)
- **Contact/Joint/Interface State** (Penetration $\Delta_n \leq 0$)

In equation 104.560, if the penalty stiffness parameter $k_n$ is assumed to be constant and independent of penetration ($\Delta_n$), it is referred as a hard contact. This type of contact/interface is more physical for interactions between two rigid surfaces or bodies. However, to model interaction between soft-soil and rigid foundation, a soft contact/interface with penalty stiffness increasing with penetration is preferred. The following Section 104.7.2.2 and 104.7.2.3 describes hard and soft contact/interface respectively.

For coupled contact/interface problems described in Section ??, in order to enforce the no-drainage condition in contact/interface normal direction between $U$ (soil) and $u$ (foundation) degrees of freedom, an additional penalty stiffness parameter $k_p$ is required. Section ?? describes how penalty stiffness $k_p$ is used to enforce the undrained condition.

It must be noted that in the penalty or any other method (Lagrange, barrier .. etc.) as described in (Wriggers, 2002), to get to the solution it has to take into account of whether the contact/interface is active or not. Thus, the inequality constraint has to be changed to the active (closed gap) or inactive (open gap) based on the state of contact. As a result, the topology of the structure changes due to
contact. This points out one of the difficulties while solving the contact/interface problem i.e. the stiffness matrix changes with active or inactive constraint equations.

As compared to one of the popular Lagrange method, the penalty method leads to non-physical penetration but does not create any additional variables. However, the non-physical penetration could be utilized to model more complicated normal contact/interface force function as such for soft contact/interface shown in Figure ?? and non-linear shear interface models as described in section ??.
104.7.2.2 Hard Contact/Joint/Interface

In hard contact, the normal penalty stiffness $k_n$ is assumed to be constant with penetration $\Delta_n$. As a result, the normal contact/interface force $F_n$ or stress $\sigma_n$ varies linearly with penetration.

$$
F_n = k_n \Delta_n \\
\sigma_n = k_n \epsilon
$$

(104.562)

where $k_n$ represents the normal stiffness between soil-structure interface and $\Delta_n$ is the penetration in contact/interface normal direction.

Figure 104.35: Hard contact/interface normal (a) force and (b) stiffness function with penetration $\Delta_n$

Figure 104.35 shows the normal force $F_n$ or stress $\sigma_n$ and stiffness $k_n$ as a function of penetration $\Delta_n$ or normal strain $\epsilon$ respectively. The normal stiffness $k_n$ is assumed to be constant and thus has an abrupt jump or discontinuity at $\Delta_n = 0$ leading to $C^0$ continuity. The abrupt change of stiffness could often lead to numerical convergence problems.
104.7.2.3 Soft Contact/Joint/Interface

For rocks, (Gens et al., 1990) presented a nonlinear (hyperbolic) function of elastic normal stress with penetration. The function had different stiffness for loading and unloading up to a permanent deformation $u_{mc}$. The hyperbolic function $u/(u - u_{mc})$ has a singularity at $u = u_{mc}$. It does not increase monotonically and does not possess continuous derivatives for $u \geq 0$. In FEM since, the stiffness cannot be infinite at $u = u_{mc}$ and the displacement $u$ can be greater than $u \geq u_{mc}$, which can lead to numerical instability and convergence problems. As per authors knowledge, there has not been enough experimental investigation to understand the normal contact/interface behavior of the soil-structure interface. Desai and Nagaraj (1988) claim to have performed cyclic normal tests on a concrete-soil interface on a shear box but did not show any experimental results. Desai and Nagaraj (1988) idealized the normal behavior based on the critical state soil mechanics as shown in Figure 104.29. (Bandis et al., 1983) investigated the response of fresh and weathered rock. It could be observed that after some $(2-4)$ cycles, the loading and unloading curve fairly overlaps and could be approximated by the same function.

The normal behavior at the interface being modeled here using penalty stiffness approach, a non-linear exponential elastic function is assumed for the soft contact. The parameters include an initial stiffness $k_i$ and a stiffening rate $S_r$ to control the normal stress $\sigma_n$ function with penetration $\Delta_n$. In comparison to hard contact, soft contact/interface thus has a smooth exponential, normal contact/interface force function with penetration as shown in Figure ??.

![Figure 104.36: Soft contact/interface normal (a) force and (b) stiffness function with penetration $\Delta_n$.](image)

The exponential variation is expected to match the realistic increasing contact force with penetration as shown in Figure 104.29. As stated earlier, it must be noted that in this model the response is assumed to be fully non-linear elastic with no tensile region. As a result, the loading and unloading stiffness and response is the same.
The non-linear normal force $F_n$ or stress $\sigma_n$ is defined as:

\[
F_n = k_i \exp(-S_r \Delta_n) \Delta_n \\
\sigma_n = k_i \exp(-S_r \epsilon) \epsilon
\]  

(104.563)

where $k_i$ represents the initial normal stiffness between soil-structure interface and $S_r$ represents the stiffening (or hardening) rate with penetration $\Delta_n$ or normal strain $\epsilon$. Equation (104.563) represents the normal force $F_n$ and stress $\sigma_n$ for force based and stress based contact respectively. At soil-foundation interface, as the foundation penetrates more, the soil becomes harder resulting in an increase of interface normal stiffness $k_n$ and normal stress $\sigma_n$. The stiffening rate leads to an exponential increment of contact/interface stress per unit of penetration $\Delta_n$. From the above formulation it must be noted that for the stress based contact, the penetration $\Delta_n$ is replaced with normal strain $\epsilon$. Subsequently, the parameters initial normal stiffness $k_i$ and stiffening rate $S_r$ should also be calibrated accordingly. Similarly, the stiffness and other derivatives could be obtained by replacing penetration $\Delta_n$ with normal strain $\epsilon$. The normal stiffness $k_n$ has unit of $n/m$ for the force based contact/interface and $Pa$ for the stress based contact.

Equation (104.563) could be differentiated to get the stiffness $k_n$ as:

\[
k_n = k_i \exp(-S_r \Delta_n)(1 - S_r \Delta_n) \\
k_n = k_i \exp(-S_r \epsilon)(1 - S_r \epsilon)
\]  

(104.564)

It can be observed from equation (104.564) that putting $\Delta_n = 0$, the normal stiffness $k_n$ becomes equal to initial normal stiffness $k_i$ i.e. $k_n = k_i$. When $\Delta_n \geq 0$, the stiffness grows exponentially. Extending Equation (104.563) and (104.564) to uplift (no-contact) ($\Delta_n \geq 0$), with the assumption of small initial stiffness $k_i$, the force and stiffness function would lead to $C^1$ continuity. The $C^1$ continuity thus would lead to a smooth stiffness function even at the border of contact/interface and non-contact region resulting in quadratic convergence at global FEM level for the Newton-Raphson method. However the non-linear behavior would lead to comparatively larger number of iteration than hard contact.

It is also interesting to note that by setting the stiffening rate $S_r = 0$, hard contact/interface can be recovered i.e. $k_n = k_i$ and $\sigma_n = k_n \epsilon$. This demonstrates the generality of soft contact/interface formulation.

The exponential growth of stiffness in finite element method (FEM) can lead to numerical instability for large values. To avoid this, a maximum normal stiffness $k^{max}_n$ is applied to restrict its further growth. Figure 104.37 shows the stiffness $k_n$ function with and without a cap. The stiffness function thus can be written as:

\[
k_n = \max(k_i \exp(-S_r \Delta_n)(1 - S_r \Delta_n), k^{max}_n) \\
k^{max}_n = k_i \exp(-S_r \Delta^{max}_n)(1 - S_r \Delta^{max}_n)
\]  

(104.565)
For implementing the above Equation 104.565 in FEM, there would be a need to find $\Delta_n^{max}$ corresponding to maximum allowable normal stiffness $k_n^{max}$ in order to integrate the stiffness function $k_n$. An efficient method to get $\Delta_n^{max}$ is described in Section 104.7.2.3 below.

**Iterative Method To Find $\Delta_n^{max}$**. For soft contact/interface implementation in FEM, it would be required to find out $\Delta_n^{max}$ for a given initial normal stiffness $k_i$, stiffening rate $S_r$ and maximum normal stiffness $k_n^{max}$. Since Equation 104.565 is a nonlinear function, an iterative method is needed to get to the solution. One of the best solution search methods is the bisection method, which repeatedly bisects an assumed solution interval, choosing only one of the branches where the solution might exist. In this method, an initial guess of solution space is required.

Theoretically, a large solution space could be given but would not be computationally feasible and optimal. For the given problem, it is often desired to predict solution space to get in as fewer iterations as possible. To reach the solution optimally and efficiently, an initial guess of solution range for penetration $\Delta_n$ was found to be $[k_n^{max}/k_i, 0.5k_n^{max}/k_i]$.

### 104.7.3 Shear Contact/Joint/Interface

#### 104.7.3.1 Interface Shear Zone

At the soil-structure interface, there exists a thin shearing zone of 5-20 times the $D_{50}$ Yoshimi and Kishida (1981); Martinez et al. (2015); Dejong et al. (2006); DeJong and Westgate (2009) as shown in Figure 104.38. Since the interface constitutive models are defined in stress-strain space, the applied displacements must be converted to strains. Based on the shear zone thickness $SZ_h$, the total shear
strain $\gamma$ and incremental shear strain $\Delta \gamma$ can be calculated as

$$
\gamma = \frac{\Delta t}{SZ_h}
$$

$$
\Delta \gamma = \frac{\delta \Delta t}{SZ_h}
$$

(104.566)

where $\Delta t$ and $\delta \Delta t$ are the total and incremental shear displacement at the interface between the two soil-structure contact/interface surface.

Similarly, the normal strain $\epsilon$ and incremental normal strain $\Delta \epsilon$ can be calculated as

$$
\epsilon = \frac{\Delta n}{SZ_h}
$$

$$
\Delta \epsilon = \frac{\delta \Delta n}{SZ_h}
$$

(104.567)

where $\Delta n$ and $\delta \Delta n$ are the total and incremental penetration in contact/interface normal direction. In the interface constitutive models, the normal strain $\epsilon$ is generally also referred as volumetric strain $\epsilon_v$ Stutz (2016). The normal stress is assumed to offer confinement to the interface shear band. It must be noted that although a shear band of thickness $SZ_h$ is assumed, the interface element itself has zero thickness. The shear zone thickness $SZ_h$ is a material parameter for the interface models. The shear zone thickness can vary based on the roughness of the soil-structure interface but could be generally assumed to be around 5-20 mean particle size diameter $D_{50}$.

**Shear Contact/Joint/Interface Models.** This section describes three models to describe the non-linear shear interface behavior which is intended to capture some of the actual soil-structure interface response.

Apart from the traditional Mohr-Coulomb i.e. Elastic-Perfectly Plastic Shear (EPPS) model, two additional non-linear models have been proposed. The Non-Linear Hardening Shear (NLHS) is a non-linear Armstrong-Frederick type hardening model where the normalized shear stress parameter $\mu = \tau / \sigma_n$...
increases non-linearly from 0 to residual normalized shear stress $\mu_r = \tau_r / \sigma_n$. Non-Linear Hardening Softening Shear (NLHSS) adds one more level of sophistication. It can model the softening of normalized shear stress parameter $\mu$. Once the peak normalized shear stress $\mu_p = \tau_p / \sigma_n$ is attained, it starts to decrease to the residual normalized shear stress $\mu_r$. Figure 104.39 shows a typical monotonic response of the three models for a constant normal stress $\sigma_n$.

![Figure 104.39: Comparison of the interface models with monotonic response](image)

From the Figure 104.39, it can be observed that the models behave quite differently. The EPPS model reaches the residual state at very small shear strain $\gamma = \Delta u / S Z_h$ level where as because of non-linear hardening both NLHS and NLHSS reach at larger shear strains $\gamma$. EPPS and NLHS do not show any peak behavior whereas NLHSS shows a peak followed by softening to residual strength. The models are explained in detail in the next section. NLHS and NLHSS model assume to have negligible to no elastic region and follow the elasto-plasticity theory. And since, the linear Armstrong-Frederick hardening parameter $H_a$ is assumed to be equal to the elastic stiffness $k_t$ resulting in the overall elasto-plastic stiffness equal to $0.5k_t$. The thin shear zone at the soil-structure interface starts to develop plastic deformation as soon as a small shear stress $\tau$ is applied. It must also be noted in Figure 104.39 that the elastic shear stiffness $k_t$ depends upon the normal stress $\sigma_n$. Thus, for a given normal stress $\sigma_n$, the shear stiffness $k_t$ is defined as

$$k_t = k_t^{\sigma_n} = (k_t)^{\sigma_n_0} \frac{\sigma_n}{\sigma_{p0}}$$

(104.568)

where $\sigma_{p0}$ is the constant stress of 101.3kPa and $(k_t)^{\sigma_n_0}$ is the shear stiffness for a normal stress of $\sigma_n = \sigma_{p0} = 100kPa$. The models are thus developed using kinematic hardening plasticity with initial
kinematic hardening stiffness $H_a$ equal to the elastic stiffness $k_t$. As stated above, this leads to initial elastic-plastic stiffness equal to $0.5k_t$ and also results in incremental elastic strain energy equal to the incremental plastic free energy as described in Section ??.

### 104.7.3.2 Elastic Perfectly Plastic Shear (EPPS) Model

The simplest shear interface model is the Mohr Coulomb interface model with an elastic stiffness under no slippage and zero stiffness when it slips. The material behavior is of type elastic-perfectly plastic type. The yield function ($f$) is thus given as

$$f := \tau - \mu \sigma_n \leq 0 \quad (104.569)$$

where $\mu$ is a constant coefficient of friction, $\tau$ is the shear stress and $\sigma_n$ is the normal stress.

Figure 104.40 shows the performance of EPPS model for different loading conditions. Since it is a elastic perfectly-plastic model, the shear stiffness $k_t$ is constant with shear strain $\gamma$ or displacement $\Delta t$ and becomes zero (perfectly-plastic state) when it reaches its residual friction coefficient $\mu_r$. Figure 104.40(a) and 104.40(c) shows the monotonic and full cyclic response with elastic perfectly-plastic behavior respectively. Figure 104.40(b) and 104.40(d) shows the monotonic cyclic behavior before and after reaching the residual friction coefficient $\mu_r$. This kind of interface behavior is mostly observed between rigid surfaces in contact. For more realistic soil-structure interface non-linear yield function should be used as described in the coming sections.

### 104.7.3.3 Nonlinear Hardening Shear (NLHS) Model

In this model, the normalized shear stress hardening variable $\mu$ increases from 0 to critical or residual normalized shear stress $\mu_r$, using the non-linear Armstrong Frederick type hardening law. The evolution rule for frictional hardening variable $\mu$ is given as

$$\Delta \mu = k_t \Delta \gamma^P - \frac{k_t}{\mu_r} |\Delta \gamma^P| \mu \quad (104.570)$$

where $k_t$ is the non-linear elastic hardening variable and $\Delta \gamma^P$ is the plastic part of the shear strain $\Delta \gamma$. The material behavior is thus of type non-linear hardening type.

The yield function ($f$) is still given as

$$f := \tau - \mu \sigma_n \leq 0 \quad (104.571)$$

where the normalized shear stress hardening variable $\mu$ evolves by Equation 104.570, $\tau$ is the shear stress and $\sigma_n$ is the normal stress.
Figure 104.40: Response of Linear Elastic Perfectly Plastic Shear (EPPS) Model with normal stress of 100 kPa, residual coefficient of friction $\mu_r = 0.68$, shear stiffness $k_t = 200 kPa$ and shear zone length $SZ_h = 5 mm$.

Figure 104.41 shows the performance of NLHS model for different loading conditions. It can be observed that the hardening variable $\mu$ increases non-linearly from 0 to residual normalized shear stress ratio $\mu_r$ at large shear displacements. As compared to the EPPS models, it is more realistic as the soil-structure interface develops the shear strength gradually with increments of shear strain $\Delta \gamma$ or shear displacements $\Delta t$.

For loose or low relative density $D_r$ soil at soil-structure interface Fakharian and Evgin (1996); Shahrour and Rezaie (1997) as shown in Figure 104.31(b) and 104.33(b), this model could be calibrated to model the non-linear hardening response. The monotonic and full cyclic response of this model shown in Figure 104.41(b) can be seen to match the interface behavior investigated by Uesugi et al. (1989) and is shown in Figure ???. Figure 104.41(d) shows the response of the model subject to cyclic shearing before reaching the residual strength. It could be observed that it is able to model the non-linear interface...
behavior as investigated by Fakharian and Evgin (1996); Shahrour and Rezaie (1997) which is shown in Figure 104.32 and Figure 104.33(c) respectively.

For dense soil with higher relative density $D_r$, it is important to model the peak normalized shear stress $\mu_p$, followed by the softening behavior until the residual shear stress $\mu_r$ is reached. The NLHSS model proposed in next section can be used to model both hardening and softening.

\[ \text{Normalized Shear } \frac{\tau}{\sigma} \]

\[ \text{Shear Displacement [mm]} \]

\[ \text{Normalized Shear } \frac{\tau}{\sigma} \]

\[ \text{Shear Displacement [mm]} \]

\[ \text{Normalized Shear } \frac{\tau}{\sigma} \]

\[ \text{Shear Displacement [mm]} \]

\[ \text{Normalized Shear } \frac{\tau}{\sigma} \]

\[ \text{Shear Displacement [mm]} \]

Figure 104.41: Response of Non-Linear Hardening Plastic Contact/Joint/Interface (NLHS) model with normal stress of 100kPa, residual normalized shear stress of $\mu_r = 0.68$, shear stiffness $k_t = 400kPa$ and shear zone length $SZ_h = 5mm$.

104.7.3.4 Nonlinear Hardening Softening Shear (NLHSS) Model

In this model, the normalized shear stress hardening/softening variable $\mu$ increases from 0 to its peak limit $\mu_p$, and then with more shear displacement reaches to the residual normalized shear stress of $\mu_r$ using a non-linear Armstrong Frederick type hardening/softening law. The evolution of the hardening/softening
variable $\mu$ during hardening phase is given as

$$\Delta \mu = k_t \Delta \gamma^p - \frac{k_t}{\mu_p} |\Delta \gamma^p| \mu$$  \hspace{1cm} (104.572)

where $k_t$ is the non-linear hardening variable and $\Delta \gamma^p$ is the plastic part of incremental shear strain $\Delta \gamma$. Once the peak normalized shear stress is attained, the material starts to soften. The softening behavior is modeled as reduction of normalized shear stress ratio $\mu$ as

$$\Delta \mu = -\frac{n \ast b(\mu_p - \mu_r)}{\left(\frac{\pi}{2}\right)^n \theta^{1/n-1}} \ast cos^2 \theta \Delta \gamma^p$$ \hspace{1cm} (104.573)

$$\theta = \frac{\mu_p - \mu}{\mu_p - \mu_r} \left(\frac{\pi}{2}\right)^n$$ \hspace{1cm} (104.574)

where $b$ is the softening rate, $\Delta \gamma^p$ is the incremental plastic shear strain and $n$ represents the size of the peak plateau as shown in Figure 104.42. This incremental form of softening phase is derived from the inverse tangent function raised to power $n$ as

$$f = a \ast (\arctan(b \ast \gamma^P))^n$$ \hspace{1cm} (104.575)

where $a$ is a constant parameter of the function and in Equation 104.574 is equal to $(\mu_p - \mu)/(\mu_p - \mu_r)$. The softening rate parameter $b$ represents the rate at which the normalized shear stress $\mu_p$ decreases with further application of shear displacement $\Delta \tau$ as shown in Figure 104.42. A larger value of $b$ would result in faster decay. The size of peak plateau parameter $n$ determines the size of the plateau formed at the peak as shown in Figure 104.42. A larger value of $n$ would result in a larger plateau. It must be noted that the peak plateau size parameter $n$ also influences the overall rate of softening as could be
seen from the derived incremental Equation 104.574. The parameters $n$ and $b$ can be calibrated from monotonic shearing tests. Figure 104.43 shows the response of the model with the peak plateau size parameter $n = 4$ and softening rate parameter $b = 40$.

The yield function ($f$) is again given as

$$f := \tau - \mu \sigma_n \leq 0$$  \hspace{1cm} (104.576)

where $\mu$ is a normalized shear stress hardening/softening variable evolved by Equation 205.17 and 104.574, $\tau$ is the shear stress and $\sigma_n$ is the normal stress.

Since in the model, the hardening law is defined as Armstrong-Frederick type, the peak shear stress ratio $\mu_p$ is defined as only 95% of the asymptotic limit of the Equation 205.17. Thus, the asymptotic limit of Armstrong-Frederick type hardening is raised by a factor of $1/0.95$ times the peak shear stress ratio $\mu_p$.

In this model during cyclic shearing, it is assumed that once the peak strength is passed, the material would not be able to again attain the peak strength during cyclic loading. It is based on the assumption that as the material passes the peak strength, particle breakage and smoothening of the surface takes place which cannot be recovered back by any process. Thus as the material softens, the peak coefficient of friction $\mu_p$ is iteratively redefined to the coefficient of friction $\mu$ in the softening phase. This effect can also be observed from the tests performed by Uesugi et al. (1989) as shown in Figure ??(b). As soon as the peak shear stress ratio $\mu_p$ degrades to residual strength in the first cycle, other cycles follow the residual shear stress ratio $\mu_r$.

Figure 104.41 shows the performance of NLHSS model for different loading conditions. The response of the model is very close to the realistic interface behavior observed by Uesugi et al. (1989) as shown in Figure ??(b). During cyclic shearing shown in Figure 104.41(c), the model predicts the peak behavior only in the first cycle. After that, the response is governed by the residual normalized shear stress $\mu_r$. Figure 104.41(c) shows the cyclic shearing behavior when the residual normalized shear stress $\mu_r$ is not attained in the first cycle. As a result of unloading and reloading, it again attains the last peak normalized shear stress $\mu_p$ that it had attained during the softening phase.

NLHSM can be extended further to model the variation of peak normalized shear stress $\mu_p$ for different normal stress $\sigma_n$ conditions as observed in tests by Fakharian and Evgin (1996); Shahrou and Rezaie (1997); Evgin and Fakharian (1997); Fakharian and Evgin (1997); Fakharian (1996); Fakharian et al. (2002). For a given relative density $D_r$ of soil in the sheared zone, the peak normalized shear stress $\mu_p$ can be generalized to be a logarithmic function of normal stress $\sigma_n$ as

$$\mu_p = \mu_{p_0} - k \log(\sigma_n/P_0)$$  \hspace{1cm} (104.577)
Figure 104.43: Response of Non-Linear Hardening Softening Shear (NLHSS) model with normal stress of 100kPa, residual normalized shear stress $μ_r = 0.68$, peak normalized shear stress $μ_p = 0.9$, shear stiffness $k_t = 800kPa$, peak-plateau parameter $n = 4$, softening rate parameter $b = 40$ and shear zone length $SZ_h = 5mm$.

where $μ_{p_0}$ is the peak normalized shear stress at normal stress of $σ_n = P_0$, $k$ is the peak normalized shear stress rate of decrease and $P_0$ is the reference stress of $P_0 = 100kPa$. This is similar to the Bolton (1986) stress-dilatancy relationship observed in sands. In the above Equation 104.577, for the normal stress of $σ_n ≤ P_0$, the peak normalized shear stress $μ_p$ would become greater than $μ_{p_0}$ as the term $log(σ_n/P_0)$ becomes negative. This would result in high peak normalized shear stress $μ_p$ for low confining stress $σ_n$. As a result, the peak normalized shear stress $μ_p$ needs to be restricted to a value. In this model, it is assumed that $μ_{p_0}$ would also act as the maximum possible peak normalized shear stress $μ_p$ for low normal stresses or confinement. Thus the above Equation 104.577 can be re-written...
with the limit on peak normalized shear stress $\mu_p$ as

$$\mu_p = \max(\mu_{p0}, \mu_{p0} - k \log(\sigma_n/P_0)) \quad (104.578)$$

where $\mu_{p0}$ also represents the maximum peak normalized shear stress $\mu_{p}^{\max}$, that the sheared zone soil could attain. Thus, with experiments conduction for different normal stress $\sigma_n$, the peak normalized shear stress $\mu_p$ can be calibrated as the function of normal stress $\sigma_n$ with peak normalized shear stress limit $\mu_{p0}$ and peak normalized shear stress rate of decrease $k$. Figure ?? shows the response of the model for different normal loads of 100$\text{kPa}$, 250$\text{kPa}$ and 500$\text{kPa}$ and also validates the model with the experimental results from Fakharian and Evgin (1996) as shown in Figure 104.31.

### Extending The Models to 3D

Section ?? described the model in 2-D in $\tau - \sigma_n$ space, to make the reader understand the basics of the model. The model can be easily extended to 3-D using the back stress variable $\alpha$ instead of the normalized shear stress $\mu$ for NLHS and NLHSS models. In 3-D, there would be normal stress component $\sigma_n$ in contact/interface normal direction $n$ and two tangential stress components $\tau_1$ and $\tau_2$ in tangential contact/interface plane in directions 1 and 2 respectively. Similarly, the shear strain $\gamma$ has two components as $\gamma_1$ and $\gamma_2$ in the two tangential directions 1 and 2 respectively. As stated earlier, since normal interface behavior is assumed to be non-linear elastic, the plastic strains are only developed in shear. Thus the incremental plastic strain $\Delta \gamma^p$ and its magnitude $|\Delta \gamma^p|$ in these models is defined as

$$|\Delta \gamma^p| = \sqrt{\Delta \gamma_1^p \Delta \gamma_2^p} \quad (104.579)$$

$$\Delta \gamma^p = [\Delta \gamma_1^p, \Delta \gamma_2^p] \quad (104.580)$$

where $\Delta \gamma_1^p$ and $\Delta \gamma_2^p$ represents the incremental plastic shear strain components in contact/interface tangential (shear) direction 1 and 2 respectively. The plastic flow direction $m$ is defined as the direction of incremental plastic shear strain $\delta \gamma^p$ and thus is defined as $m = \delta \gamma^p / |\Delta \gamma^p|$.

### 104.7.3.5 EPPS Model

EPPS Model described in Section 104.7.3.2, is the simplest Mohr Coulomb type elastic perfectly-plastic model. The yield function $(f)$ is defined as

$$f := (\tau_1/\sigma_n - \mu_r)^2 + (\tau_2/\sigma_n - \mu_r)^2 = 0 \quad (104.581)$$

where $\sigma_n$ is the normal stress, $\tau_1, \tau_2$ are the shear stress and $\mu_r$ is the residual friction coefficient. Being an elastic-perfectly plastic model with no internal variables, there is no hardening/softening evolution rule.
104.7.3.6 NLHS Model

The non-linear hardening shear model described in Section 104.7.3 is modeled in 3-D using the back stress internal variable $\alpha$. The yield function ($f$), is defined as

$$ f := \left( \frac{\tau_1}{\sigma_n} - \alpha_1 \right)^2 + \left( \frac{\tau_2}{\sigma_n} - \alpha_2 \right)^2 = 0 $$

(104.582)

where $\sigma_n$ is the normal stress, $\tau_1, \tau_2$ are the shear stress and $\alpha_1, \alpha_2$ are the back stress components in contact/interface tangential direction 1 and 2 respectively on the contact-interface plane. The hardening law would be then defined as

$$ \Delta \alpha = k_t \Delta \gamma^p - \frac{k_t}{\mu_p} |\Delta \gamma^p| \alpha $$

(104.583)

$$ \Delta \alpha = H_m |\Delta \gamma^p| $$

(104.584)

$$ H_m = k_t m - \frac{k_t}{\mu_p} \alpha $$

(104.585)

where $\mu_p$ is the peak normalized shear stress, which depends upon the normal stress $\sigma_n$ as stated in Equation 104.577 and $k_t$ represents the initial elastic shear stiffness of soil-structure interface. $H_m$ represents the non-linear Armstrong-Frederick type hardening modulus and $m$ represents the plastic flow direction.

104.7.3.7 NLHSS Model

The non-linear hardening softening shear model described in Section 104.7.3.4 is also modeled in 3D using the back stress internal variable $\alpha$. The yield function ($f$) is defined as Equation 104.582. The hardening evolution law for the back stress $\alpha$ is defined by Equation 104.585. The softening law is defined as Von-Mises type as

$$ \Delta \alpha = -n^* b(\mu_p - \mu_r) \frac{\pi}{2^{n+1}} \frac{\cos^2 \theta \Delta \gamma^p}{n^*} $$

(104.586)

$$ \theta = \frac{\mu_p - |\alpha|}{\mu_p - \mu_r} \frac{\pi}{2^n} $$

(104.587)

$$ \Delta \alpha = S_m \Delta \gamma^p $$

(104.588)

$$ S_m = -n^* b(\mu_p - \mu_r) \frac{\pi}{2^{n+1}} \cos^2 \theta $$

(104.589)

where $\mu_r$ is the residual normalized shear stress that is constant and depends upon the soil and structure material, $\mu_p$ is the peak normalized shear stress at the start of softening phase and $S_m$ represents the softening modulus.
As stated in Section 104.7.3.4, for NLHSS model, the peak normalized shear stress $\mu_p$ of the material is iteratively defined to the back stress $\mu_p = |\alpha|$ in softening phase. This means that the dilatancy surface squeezes towards the critical surface as the sheared zone soil at interface continues to shear.

Using the incremental Equation 104.585 Equation 104.589 presented for both hardening and softening phase respectively, the 3-D model can be integrated. Using the elastic-plastic theory Hill (1950); Temam (1985); Wu Tai (1966); Lubliner (1990); de Borst and Feenstra (1990); de Borst (1987). The elastic-plastic stiffness or consistent tangent stiffness Jeremić (1994); Crisfield (1987) can then be computed easily at each loading increment or iteration. Section ?? presents the elastic-plastic integration for the interface models.

### 104.8 Inelastic Behavior and Models for Rock

#### 104.8.1 Overview of Intact Rock Behavior

Rock, and other geomaterials, feature a distinct set of material behavior that separates them from other natural and/or man made materials. Rock material subjected to shock waves in particular shows a variety of response regimes and behavior that warrant further discussion. Of particular interest are the following specifics of rock behavior: pressure sensitivity, dilative and compactive response, inherent and induced anisotropy, full coupling of porous rock solid with pore fluid and temperature fields and bifurcation response, resulting in shear and compaction bands. Each of these aspects of rock behavior will be described in sections below.

It is also very important to note that this study will focus on behavior of intact rock material, while main focus of research in rock mechanics is on behavior of jointed rock masses. While behavior of discontinuous, jointed rock mass is very important for construction in rock (tunnels, dams, foundations...), behavior of intact rock mass becomes very important for any modeling and simulation of deeper rock structures, particularly where strong shock waves are involved. This is emphasized by the fact that rock blocks do behave like solid for high pressures. For example, during shock loading blocks might fuse at contact/interface and behave like a solid, in which case the inelastic (elastic-plastic) behavior of intact rock mass has major impact on overall rock mass response.

This section uses selection of published results to emphasize distinct features of mechanical response of rock that are considered important for proper modeling of shock wave propagation. While there exist a significant body of published work on behavior of intact rock (which is still much smaller than body of published work on behavior of rock as discrete media), chosen here are publications and results that provide important results used to emphasize distinct features of mechanical behavior of rock. Emphasizing these distinct feature of rock behavior is important for a number of reasons. Two main reasons...
are that while the shock wave theory was developed over last two centuries (see brief overview in section 104.8.3) main focus was on ideal domains (linear elastic solids or fluids), rock features many distinct modes of mechanical response that demand use of high fidelity numerical modeling. In addition to that, even when shock wave theories were developed for inelastic (elastic-plastic) solids, this was done for metals, elastic-plastic response of which lacks many features found in rock (geomaterial) response (pressure sensitivity, anisotropy, dilative and compactive response...).
104.8.1.1 Pressure Sensitivity

Rock material response shows strong pressure sensitivity. Both initial yielding response, inelastic response and ultimate strength strongly depend on confinement pressure experienced by rock material. Confinement pressure acting on rock can be inherent, coming from location of rock mass (depth) and from geologic factors (tectonics). Fairhurst (2003) details initial stress determination procedures. Initial stress determination (stress level from previous, historical loading stages, tectonics, erosion) is very important in view of pressure sensitivity and elastic-plastic response of rock. Depending on the type of rock material, the effects of confinement pressure on response vary. Figures 104.44 – 104.52 show rest data for a full elastic-plastic response of various rock specimens for varying confinement pressures. Both axial ($\epsilon_1$) and radial ($\epsilon_2$) strains are shown versus axial stress ($\Delta \sigma_1$). It is very important to recognize a number of distinct features of rock behavior that these test show:

- Varying confinement pressure influences yield stress, that is an increase of yield stress is observed with the increase in confinement pressure,

- Elastic stiffness increases with increase in confinement pressure,

- Ductility increases with increase in confinement pressure,

- Increase in confinement pressure significantly influences final strain level achieved

- Volume change (dilation, as observed from radial strain results ($\epsilon_2$) is significant, however, in some cases it increases and in some decreases with confinement pressure increase (as observed in Figures 104.44 – 104.49)

- Elastic degradation/damage is increasing with cycles of loading (Fig. 104.51 for example, reduction of loading-unloading-reloading slope signifies elastic damage)

- Presence of energy dissipation in rock for various confinement pressures (area of hysteresis loops, see for example Fig. 104.50 – 104.52)
Figure 104.44: Full elastic-plastic response of marble specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) and radial ($\epsilon_2$) strains versus axial stress ($\Delta\sigma_1$) for triaxial loading of 3D samples (Stavrogin et al., 2001).

Figure 104.45: Full elastic-plastic response of lignite specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) and radial ($\epsilon_2$) strains versus axial stress ($\Delta\sigma_1$) for triaxial loading of 3D samples (Stavrogin et al., 2001).
Figure 104.46: Full elastic-plastic response of granite specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) and radial ($\epsilon_2$) strains versus axial stress ($\Delta\sigma_1$) for triaxial loading of 3D samples (Stavrogin et al., 2001).

Figure 104.47: Full elastic-plastic response of sandstone (non-burst prone) specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) and radial ($\epsilon_2$) strains versus axial stress ($\Delta\sigma_1$) for triaxial loading of 3D samples (Stavrogin et al., 2001).

Figure 104.48: Full elastic-plastic response of sandstone (burst prone) specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) and radial ($\epsilon_2$) strains versus axial stress ($\Delta\sigma_1$) for triaxial loading of 3D samples (Stavrogin et al., 2001).
Figure 104.49: Full elastic-plastic response of sulphidic ore specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) and radial ($\epsilon_2$) strains versus axial stress ($\Delta \sigma_1$) for triaxial loading of 3D samples (Stavrogin et al., 2001).

Figure 104.50: Full elastic-plastic response, with loading-reloading cycles of Periodite (upper) and Diorite (lower) specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) strains versus differential stresses ($\sigma_1 - \sigma_2$) for triaxial loading of 3D samples (Mogi, 2006).
Figure 104.51: Full elastic-plastic response, with loading-reloading cycles of Trachite (left) and Marble (right) specimens for varying confinement pressures. Shown are axial ($\epsilon_1$) strains versus differential stresses ($\sigma_1 - \sigma_2$) for triaxial loading of 3D samples (Mogi, 2006).
Figure 104.52: Full elastic-plastic response, with loading-reloading cycles of Tuff specimens (two different rock sources) for varying confinement pressures. Shown are axial ($\epsilon_1$) strains versus differential stresses ($\sigma_1 - \sigma_2$) for triaxial loading of 3D samples (Mogi, 2006).
104.8.1.2 Dilative and Compactive Response

Rock (similar to other geomaterials) feature dilative (increase in volume) and compactive (decrease in volume) response for both hydrostatic confinement as well as for deviatoric loading (shear stresses). Lockner and Stanchits (2002) performed a number of tests on (initially isotropic) sandstone and measured the undrained poroelastic response for changes in mean (hydrostatic, normal) and deviatoric (shear) stress. While change in pore pressure was found to result from changes in mean (hydrostatic) stress, it was also resulting from changes in deviatoric stress. This dependence of pore pressure, and consequentially pore volume, on deviatoric stress is called dilatancy and is a feature found in most geomaterials. Figure 104.53 shows measured dependence of a coefficient $\eta$, defined as a ratio of change in pore pressure due to change in deviatoric stress ($\Delta p = -\eta \Delta \sigma_{\text{deviatoric}}$).

![Figure 104.53: Measured poroelastic (Skempton’s) coefficient B and $\eta$ for Berea Sandstone. (Lockner and Stanchits, 2002).](image)

Figure 104.53 also shows measured values for coefficient $B$, which defines a ratio of change in pore pressures due to change in mean stress ($\Delta p = -\eta \Delta \sigma_{\text{mean}}$). Coefficient $B$ and $\eta$ are also known as Skempton’s coefficients (Skempton, 1954). Hamiel et al. (2005) developed a model based on data provided by Lockner and Stanchits (2002) that captures dilatancy in poroelastic regime. In addition to
that, one of their models allows for variability in Skempton’s coefficients and seems to capture test data quite well. While presented development focuses on seismic pore fluid pressure development (positive and negative) it has modeling and simulation implications for other dynamic, rock events where pores are filled with fluid and dilatancy is involved (which is always the case).

Dilatant response in elastic-plastic regime can be observed in Figures 104.44 – 104.49. In particular, volumetric strain calculations \( \epsilon_{\text{volumetric}} = \epsilon_{ii} = \epsilon_1 + 2\epsilon_2 \) reveal that dilatant response is present during elastic phase of loading (before yielding), significant dilation occurs after plastic limit. Thus we can conclude that while dilatancy is present in elastic phase of loading (as concluded by Lockner and Stanchits (2002)) dilatancy is even more pronounced in elastic-plastic loading regime.

In the limit of compactive and dilative response is the localized compactive/explosive deformation. Olsson (1999, 2001) details recent findings of compaction bands. They are thin zones of pure compressional deformation with very low permeability and porosity. Porosity drop for such zones is on the order of ten times \((10 \times)\) when compared to porosity of surrounding rock (Olsson, 1999). In addition to that, compaction zones (bands) growth is described in terms of shock wave analysis. Issen and Rudnicki (2001) developed a more general theory of compaction band formation, particularly in view of plasticity models with cap. Plasticity models with cap are necessary if realistic behavior of rock is to be modeled, covering a wide range of stress states, from tension, shear to compression. Borja (2006) reviews analytical conditions for appearance of volume implosion/explosion (diffuse process) and compaction bands (localized process). He shows that general conditions for localization of deformation can be used for both shear as well as for compaction and dilative localization of deformation.
104.8.1.3 Anisotropy

Various rock types feature directional features (bedding, foliation, flow structures) which are reflected in anisotropic elastic and plastic properties. Anisotropy of rock behavior can be inherent and/or induced. Inherent anisotropy is present before the current loading is applied and is most likely resulting from past geological processes in rock (past loading). Induced anisotropy results from current loading processes and can significantly change orientation and value of elastic constants from inherent values (Amadei and Goodman, 1982; Amadei, 1983). Anisotropy can also be apparent, when bedding planes obvious and hidden when bedding planes not directly observable. Measurements of anisotropy are not readily available in literature as such tests are quite involved and complex. For example Pariseau (2006) reports that measurements indicate that Young’s modulus is parallel to bedding plane is often about twice as large as Young’s modulus perpendicular to bedding plane. Figure 104.54 shows usual test setup when rock cores are extracted with different orientations to the test loading. For such anisotropic test setups, variations of axial Young’s modulus are shown in Figure 104.55.

![Figure 104.54](image)

Figure 104.54: Test setup for measuring elastic anisotropy for rock cores extracted at different orientations (Pariseau, 2006).

In addition to influence of anisotropy on elastic constants, permeability is significant influenced by anisotropy of rock. A.Angabini (2003) shows significant influence of rock anisotropy on both elastic properties and on anisotropic permeability. He used 438 samples with distinct orientations to measure isotropic and anisotropic material properties out of research wells in The Netherlands along of 2.6 km test section and going to depths of up to 1.6 km. His measurements indicate that in addition elastic anisotropy, permeability anisotropy is significant. Figure 104.56 shows differences (variation) in measured specific permeabilities at different depths for wells in The Netherlands.

In addition to significant anisotropy in permeability (for example at depth 345m permeability in...
Figure 104.55: Anisotropic variations of axial Young's moduli for different orientation of core samples (Pariseau, 2006).

Figure 104.56: Measured specific permeabilities at different depths for wells in The Netherlands, parallel to bedding \((x, y)\) and perpendicular to bedding \((z)\) (A. Angabini, 2003).

vertical direction \((Z)\) is three times higher than the one in \(X\) and 5 times higher than permeability in \(Y\) direction), significant spatial variation of permeability is present. For example, within 6 meters (between depth of 339m and 345m difference is two orders of magnitude!

Measured anisotropy of elastic and elastic-plastic properties is also significant. For example, Figure 104.57 shows directional elastic moduli (obtained using either static or dynamic tests, for dry and saturated conditions) for mudstone and sandstone.
Differences between elastic moduli in different directions of two or more are obvious. In addition to that, since elastic moduli in all three directions are different, rock exhibits general anisotropy (as opposed to cross anisotropy, where two elastic moduli would be the same).

Similar to elastic moduli (Young’s moduli), Poisson’s ratios do show significant anisotropy as shown in Figure 104.58.

In addition to elastic anisotropy (as shown in Figures 104.57 and 104.58) uniaxial strength also shows very strong directional dependence as shown in Figure 104.59.

Anisotropy of rock is not localized to one region of the domain, rather it is present in every level. Figure 104.60 shows elastic anisotropic moduli for wells in the Netherlands test area for a full depth of 1.7km. It is important to note that in addition to anisotropy that extends throughout depth. inherent uncertainty on measured moduli is present as well. This uncertainty (spread of data points for same depth) will be discussed in some more details in section 104.8.2.

In view of importance of anisotropy on rock response, determination of anisotropy is very important if simulations are to be used in predicting rock behavior. This is particularly true for modeling and simulation of shock wave propagation as anisotropy (both in elasticity and permeability) can significantly influence results.
Figure 104.58: Dynamic Poisson’s ratios for different rock samples (limestone and sandstone) in dry and saturated conditions, for wells in The Netherlands, (A. Angabini, 2003).

Figure 104.59: Uniaxial strength of mudstone for three different orientations of core sample, for various sampling depths, for wells in The Netherlands, (A. Angabini, 2003).
Figure 104.60: Anisotropic properties of mudstone and limestone at three directions at different depths (full depth profile), for wells in the Netherlands, (A. Angabini, 2003).
104.8.1.4 High Rate Elastic-Plastic Loading

Experimental data presenting (controlled) high rate elastic-plastic loading is fairly limited compared to the abundance of low rate data. It is very important to note that high rate controlled tests are of much value as they allow separation of high rate constitutive response from shock loading tests that are accomplished using high energy shock application to rock samples, which then initiate many other facets of multi-physics of rock behavior (described in more detail in section 104.8.3 on page 334). One of the best controlled high loading rate data sets were published by Stavrogin and Protosenya (1983); Stavrogin and Pevzner (1983); Stavrogin et al. (2001). A wide range of loading rates (strain rates, ranging over 12 orders of magnitude), presented (Stavrogin and Protosenya, 1983; Stavrogin and Pevzner, 1983; Stavrogin et al., 2001), allows development of qualitative understanding and development of quantitative model parameters for rock material. Figures 104.61, 104.62 and 104.63 show the influence of rate of loading on peak strength and elastic limit (yield stress) for rock samples (marble, diabase and sandstone respectively) confined to different pressures,

![Figure 104.61: Marble: dependence of peak strength (a) upper) and elastic limit (yield stress) (b) lower) on loading rate (\( \dot{\varepsilon} \)) for different confining stresses (1 – \( \sigma_2 = 0 \) Mpa; 2 – \( \sigma_2 = 20 \) Mpa; 3 – \( \sigma_2 = 50 \) Mpa; 4 – \( \sigma_2 = 100 \) Mpa; 5 – \( \sigma_2 = 150 \) Mpa) (Stavrogin and Protosenya, 1983).

While highest loading (strain) rates are on the order of \( \dot{\varepsilon} = 10^2 \) 1/s which is lower than shock loading rates (estimated to be over \( \dot{\varepsilon} = 10^5 \) 1/s) this data is still very useful as it can be used to calibrated models where strength and elastic limits are dependent or loading rate, and then use those models
Figure 104.62: Diabase: dependence of peak strength (a) and elastic limit (yield stress) (b) on loading rate ($\dot{\epsilon}$) for different confining stresses ($1 - \sigma_2 = 0$ Mpa; $2 - \sigma_2 = 50$ Mpa; $3 - \sigma_2 = 100$ Mpa; $4 - \sigma_2 = 150$ Mpa) (Stavrogin et al., 2001).

Figure 104.63: Sandstone: dependence of peak strength (full lines) and elastic limit (dashed lines) on loading rate ($\dot{\epsilon}$) for different confining stresses ($1 - \sigma_2 = 0$ Mpa; $2 - \sigma_2 = 50$ Mpa; $3 - \sigma_2 = 100$ Mpa) (Stavrogin et al., 2001).
to predict (in the sense of true prediction (Oberkampf et al., 2002)) shock wave propagation. Data presented in above Figures suggests that both peak strength and elastic limit increase with confinement pressures (pressure sensitive material, as discussed in section 104.8.1.1 on page 311). Presented data also suggest that both peak strength and elastic limit increase (linearly!) with the increase of rate of loading. Dependence of peak strength and elastic limits is proportional to loading rate increase, although such factor of proportionality is not very high. This is more evident from Figures 104.64 and 104.65 where full stress strain curves are shown for marble and granite respectively.

While data shown in Figures 104.64 and 104.65 does not cover such wide range of loading strain rates, it still leads to similar conclusion, that is increase in strain rate will lead to proportional increase in elastic limit and peak strength. Similar conclusion was drawn by Zhao et al. (1999), however, their results show much larger variation and weaker factor of proportionality.
Figure 104.64: Marble: stress-strain response for different strain rates ($\dot{\varepsilon}_1$) 1 – $\dot{\varepsilon}_1 = 2 \times 10^{-6}/s$; 2 – $\dot{\varepsilon}_1 = 2 \times 10^{-5}/s$; 3 – $\dot{\varepsilon}_1 = 2 \times 10^{-4}/s$; 4 – $\dot{\varepsilon}_1 = 2 \times 10^{-3}/s$; 5 – $\dot{\varepsilon}_1 = 2 \times 10^{-2}/s$; 6 – $\dot{\varepsilon}_1 = 2 \times 10^{-1}/s$; (Stavrogin et al., 2001).

Figure 104.65: Granite: stress-strain response for different strain rates ($\dot{\varepsilon}_1$) 1 – $\dot{\varepsilon}_1 = 10^{-5}/s$; 2 – $\dot{\varepsilon}_1 = 2 \times 10^{-4}/s$; 3 – $\dot{\varepsilon}_1 = 5 \times 10^{-2}/s$; 4 – $\dot{\varepsilon}_1 = 2 \times 10^{-1}/s$; (Stavrogin et al., 2001).
104.8.1.5 Coupling with Pore Fluid Pressure and Temperature

In addition to constitutive response of elastic, plastic and damaged matrix (solid porous skeleton) pore fluid and temperature fields have large influence on behavior of rock. Pore pressure directly influences mechanical response of solid skeleton through effective stress principle (see section 104.8.6). In addition to that, changes in temperature field will affect both pore fluid and elastic-plastic-damage characteristics of the porous solid (rock skeleton).

For example, Figure 104.66 shows dependence of stress at failure for sandstone. In particular, Figure 104.66(a) (left) shows clear dependence of failure stress on pore pressure, where increase in pore fluid pressure reduces the failure limit (stress). This is the case for different confining stresses, and the dependence is linear, which nicely follows equations that will be given in section 104.8.6. If the dependence is plotted in somewhat different form, as a function of effective stress, as shown in Figure 104.66(b) (right) all the points fall into (almost) same line. Present variation is inherent to all geomaterials and is discussed in some detail in section 104.8.2.

In addition to a full saturation, where effective stress principle is fully valid, partial saturation plays a very important role in behavior of porous rock. Partial saturation will affect response of porous rock matrix through increase in pore fluid (mix or water and air) pressures as loading is applied and rock undergoes compactive or dilatant response. Such interaction of pore fluid (mix or air and water) will also be affected by the rate of load application as the fluid viscosity starts having effects on pore pressure.

![Figure 104.66](image-url)
advection and diffusion. For example, Figures 104.67 and 104.68 show effects of moisture content on peak strength, elastic limit and coefficient of cohesion for sandstone and limestone (cohesion only). Full saturation of sandstone samples was achieved at moisture content of 3 % while for limestone it was 8 %. Results show that steady decrease of peak strength, elastic limit and coefficient of cohesion with increase in moisture content. It is very interesting to note that an increase in loading rate still increases the peak strength, elastic limit and coefficient of cohesion (as concluded in section 104.8.1.4 for dry samples), even with increase in influence of pore fluid (water and air) pressures, that now has to advect and diffuse. This type of interaction of pore fluid with loading rate and response of rock skeleton is somewhat counter-intuitive as it was expected that increase in saturation with increase in loading rate

\[ \text{Moisture content is here defined as weight percentage of the water content compared to the weight of the sample.} \]
Figure 104.68: Dependence of the coefficient of cohesion for sandstone (a, upper) and limestone (b, lower) on strain rate ($\dot{\epsilon}$) for different levels of moisture content ($W$) (Stavrogin et al., 2001).

will lead to an increase in pore fluid pressures during high loading rate events, and thus reduce peak strength, elastic limit and coefficient of cohesion.

The influence of pore fluid on response can also be followed on full stress strain curves, as shown in Figure 104.69 for limestone. While increase in water (moisture) content leads to a decrease in stiffness and elastic-plastic strength, increase in strain (loading) rate leads to an increase in stiffness and elastic-plastic strength (stress-strain curves are "higher" for faster loading). This seemingly counter-intuitive response might need to be explored in more depth.

While previously shown results assume that pore space within rock is mostly connected, Curran (1994) uses simple micromechanical models to predict that at high confinement, the behavior of fully water-saturated rock changes from a classical effective stress response (see more about effective stresses in section 104.8.6) to a much stronger and stiffer material. This transition happens at different confining pressures, depending primarily on solids bulk modulus and the pore morphology/fabric. For example, Curran (1994) note that hard rock with small ratio of crack volume to pore volume, the transition begins at 0.2 GPa, while for soft rock (or rock with high ratios of crack volume to pore volume), the transition happens at much higher confinements, of 1 – 2 GPa. This change in rock skeleton needs to be taken into account for any modeling and simulations where such threshold compressive values are reached or exceeded. In addition to that, a very important observation was made by (Larson and Anderson, 1979), in that under high confining pressures, liquid water will turn into ice VI $^{69}$ thus significantly changing the nature of coupling of pore fluid with porous solid.

Changes in temperature will affect both the pore fluid (by changing the volume and reactivity with $^{69}$Ice VI is a tetragonal crystalline form of ice formed by cooling water to 270 K at 1.1 GPa.
rock minerals), as well as the behavior of the solid (porous) matrix. With an increase in temperature, rock becomes softer but much more ductile. Inada et al. (1992) performed a number of tests on granite, andesite and sandstone, with changes of temperature from $-160^\circ C \text{ (} -256^\circ F)$ to $+100.00^\circ C \text{ (} 212^\circ F)$ for both dry and wet (saturated) rock samples. Figure 104.70 shows stress-strain curves resulting from those tests. While the influence of saturation and temperature varies for different types of rock, general trend is that with increased temperature response becomes softer (lower stiffness, lower peak strength) with higher ductility. Temperature range presented does not cover completely application area for shock loading (where temperatures might reach and exceed melting point for rock), however softening and increase in ductility trend will continue as the temperatures increase until close to melting point. At such high temperatures, rock behavior will gradually change from solid to heavy fluid with significant changes in viscosity. Holyoke and Rushmer (2002) did a number of tests on muscovite-biotite metapelites and a biotite gneiss with very high temperatures (from $650^\circ C \text{ (} 1202^\circ F)$ to $950^\circ C \text{ (} 1742^\circ F)$) and reported dilatancy effects and highly ductile response, with strains extending over 15% and reaching peak strength at 5 – essentially represents a new material that has to be properly modeled.
Figure 104.70: Stress-strain curves for dry and wet samples of granite, andesite and sandstone with varying temperature (Inada et al., 1992).
104.8.2 Uncertainty and Variability of Rock Behavior

Rock material behavior is characterized by point wise uncertainty and spatial variability. While this topic is covered in much more detail in section 506.6, given here is a brief account of experimental data that supports above statement. Test data shown in previous sections exhibits variation. This variation is present in any set of test data were more than one sample was used to determine material properties. For example, Figure 104.71 from Pariseau (2006), shows values for shear modulus for a variety of rock types. It is noted that the variability is quite large, yet this is nothing unusual and is rather characteristic of rock material. In addition to that, variability of rock material parameters is apparent in any other set of test data. For example larger and/or smaller variability is present in data presented in Figures 104.53, 104.56–104.60, 104.61–104.63, 104.66–104.68. This variability is always present (see discussion in section 506.6) and need to be taken into account in order to have higher confidence in modeling and simulation of rock behavior. One possible approach to fully incorporating uncertainty into modeling and simulation is given in section 506.6 on page 2527.

![Variation of shear modulus for different rock types (Pariseau, 2006).](image_url)
104.8.3 Effects of Shock Loading on Intact Rock Behavior

In previous sections I have attempted to present main features of intact rock behavior for a variety of loading conditions, loading rates, temperatures... Select set of publications were used to describe specifics of rock behavior, including pressure sensitivity, dilative and compactive response, anisotropy, effects of high rate of loading and effects of pore fluid pressures and temperatures and finally the uncertainty of such data. In this section, select publications on shock loading in rock are reviewed with the main aim of synthesizing previous findings and showing that all of the previous specifics of intact rock behavior influence response to shock loading.

It is very important to note that the history of development of modern theory of shock waves is long and quite interesting. Salas (2006) describes many (unsuccessful) attempts by the greatest mechanics minds of last two centuries to develop a consistent theory of shock waves. Such historic are invaluable in having the right prospective in developing consistent approaches for multi-physics modeling of shock waves in intact rock.

A very interesting set of experimental results and findings for rocks of both high porosity (tuff and limestone) (Larson and Anderson, 1979) and low porosity (granite and sandstone) (Larson and Anderson, 1980) are available in literature. For low porosity tests a number of important observation are made. Relationship of shock wave velocity versus particle velocity shows a discontinuity. It is suggested that this is due to the shear initiated pore collapse, which relates to dilatancy and compactive/shear localized deformation bands. This is indeed more appropriate, in Authors opinion, and in view of recent research on compressive localization (Olsson, 1999, 2001; Issen and Rudnicki, 2001; Borja, 2006) that such pore collapse is due to the initiation of diffusive, implosive and localized, compaction bands. In addition to that Larson and Anderson (1980) show that uniaxial strain data for low strain rates can be used to model high strain rate events, for dry rocks. This is a very important findings as it allows calibration of material models using low rate experiments for prediction of shock wave events. In addition to that, it was found that for very low porosity rock where pore are not in contact, the effective stress principle does indeed need to be used in its original form (Biot and not Terzaghi form) as given in equation 104.592 in section 104.8.6 on page 343, with \( \alpha \) having a very low value. This is consistent with findings from micromodels mentioned in section 104.8.1.5 (Curran, 1994). Test result date for highly porous rock (tuff and limestone) (Larson and Anderson, 1979) show that porosity plays a major role in behavior of rock. For example, the irreversible process of compaction and elastic unloading leads to a large hysteresis, dissipating significant amounts of wave energy, effectively damping the wave out. In addition to that, an increase of strength with strain rate is observed, reinforcing experimental observations be Stavrogin and Protosenya (1983); Stavrogin and Pevzner (1983); Stavrogin et al. (2001), described in section 104.8.1.4.
Dynamic compaction behavior of intact rock material was and still is of considerable interest. Lysne (1970) did a number of experiments on dry and water saturated tuff. The main conclusion he was able to draw was that the compaction of porous rock material is a process that is slower than the shock wave propagation and that it takes longer time (than for the compressive wave to pass) to complete such compaction process. In addition to that, Lysne (1970) was able to show, that at least for stresses in water below 2.5 GPa, the influence of heating on volume change can be neglected in porous rock saturated with water. D. Erskine and Weir (1994) presents data on dry and wet tuff, which exhibits quite complex behavior. Both pore crushing and phase change (liquefaction) are observed and are responsible for complex compression behavior. Heterogeneity of rock also plays a major role in the observed response. Another very important conclusion is that Gruneisen model (aka Debye-Grüneisen model) does not perform well, indicating that there a likely phase transition is happening and needs to be accounted for.

Hiltl et al. (1999) present interesting set of shock-recovery tests results on dry and fully saturated sandstone. Principle of effective stress is again playing important role as it is observed that water saturated samples had much smaller compaction due to distribution of confining pressures between pore water and porous solid. In addition to that, reduction of porosity due to high pressure of shock waves was much smaller for saturated samples, again proving that pore fluid carries quite a bit of load due to slow drainage, as present during shock wave loading. It is also observed that as the shock pulse duration increases, so does the damage, implying that as the pore fluid gains time to drain, effective stresses in porous solid increase and causes the damage and compaction.

### 104.8.3.1 Shock Waves of First and Second Kind

The dynamic behavior of saturated porous media was studied at length by M.A.Biot (1956); Biot (1962, 1972). In one of his studies (M.A.Biot, 1956) he concluded that there exist three kinds of coupling between pore fluid and porous solid (inertial, viscous and mechanical). He also concluded that the viscous coupling plays a key role and determines response of a coupled system to dynamic excitation, while making wave propagation dispersive. He demonstrated (analytically) the existence of two kinds of compressional waves corresponding to the mechanical and inertial coupling, while the viscous coupling is responsible for a pure wave. For high viscous coupling, the relative movements between the two phases are prevented, so there is only one compressional wave and the total mass behaves as a single-phase medium.

Recent paper by Lomov et al. (2001), actually shows an experimental proof of such coupling. Figure 104.8.3.1 shows two test results for dry and wet samples. It is important to recognize that these
test results can also be used to validate shock wave propagation modeling and simulation as described in section 104.8.6 on page ??.

![Figure 104.72: Test results for a 1D wave propagation in dry (high coupling) and wet (low coupling) sandstone (Lomov et al., 2001).](image)

For example, for high coupling case (dry sandstone, pore fluid is air), the velocity profile shown in Figure 104.8.3.1 is increasing and when integrated (to get displacements), profile corresponds (at least qualitatively) to high coupling case \( K = 10^{-6} \text{ cm/s} \) presented in Figure 309.14. Similarly, low coupling case (wet sandstone, pore fluid is water) the velocity profile increases, then drops to almost zero and then increases again. Upon integration to get displacement, this will correspond (qualitatively) to low coupling case \( K = 10^{-2} \text{ cm/s} \) presented in Figure 309.14. Permeabilities are here used qualitatively for both water and air and signify ease with which fluid (water or air) moves past porous solid. While, results presented in Figure 309.14 correspond to a linear elastic case, at least qualitatively, they follow test results, which of course correspond to elastic–plastic behavior of rock material. Such elastic-plastic behavior will affect results in many ways (slow the wave propagation speed for one), but at least it is reassuring that an elastic solution can be used to help gain understanding of the basic mechanics.

Analytical solutions for shock wave propagation, even with many simplifying assumptions, are valuable as candidates for verification and validation. Two such solutions are mentioned below. Vasilev et al. (1980) discusses interaction of gas, liquid and porous medium during and after an underground explosion. A complex interaction is described which eventually leads to the implosion of explosion cavity. Although the analysis presented is based on elastic behavior of the porous medium, it provides excellent basis for understanding phenomena involved as well as for verification. Nikolaevskiy et al. (2006) presents
more sophisticated analysis of similar phenomena, using finite difference method (Wilkins, 1999), and an elastic-plastic material model. However, Authors neglect the dilatancy of the material in order to simplify their solutions. Neglecting dilatancy certainly affects results Compaction related to coupling of pore fluid and porous solid. Moreover, influence of temperature fields is also neglected (isothermal process) which might hinder fidelity of modeling where large temperature changes are present and where temperature change influences behavior of pore fluid and porous solid.

104.8.3.2 Hugoniots

The Hugoniot curves (also known as Rankine-Hugoniot) for material present important data about material state. Material compression state defined by initial pressure, density and energy, can be used, to determine new state upon applying shock loads. Such curves prove important in material modeling for rock subjected to shock loading. Early on Afanasenkov et al. (1969) showed that it is possible to predict shock Hugoniots of any substance up to compression ratio of two with the knowledge of initial density and initial compressibility. Shipman et al. (1971) used a number of experiments to determine Hugoniots for Sandstone. In addition to that they used measured data and developed curves to show that phase boundaries do shift significantly compared to those determined using static means. This important conclusion affects development of elastic-plastic modeling for intact rock where high temperature effects cannot be neglected.
104.8.4 Material Modeling of Rock

104.8.4.1 Lawrence Livermore National Laboratory Models

A number of models originating from researchers from Lawrence Livermore National Laboratory have been developed over the years. While they do not represent a single line of development (and might have been produced by different research groups from different departments) they are summarized in this LLNL section.

Glenn (1995) presents a simple, yet effective total stress model that depends on mean confinement (pressure), temperature and on a damage parameter that serves as an internal variable and depends on degradation due to tensile and shear failure.

\[
\sqrt{3s_{ij}s_{ij}} \leq Y = (1 - D)\bar{Y} + \beta D\bar{Y}
\]

(104.590)

where \(s_{ij}\) is the deviatoric stress, \(\beta\) is a constant, and \(D\) is a scalar function of the volumetric components of void and equivalent plastic strain tensors. The generalized compressive strength \(\bar{Y}\) is a function of unconfined compressive strength, the ultimate compressive strength, the melting temperature, the mean pressure, the cohesion and an additional material constant.

More recently Lomov et al. (2001); Antoun et al. (2003), presented an elastic viscoplastic material model that takes into the account various influences on rock yielding behavior. For example, taken into the account are the effects of scaling, hardening, damage and melting. In addition to that, compaction is modeled using analytic porous compaction model, while also included are the effects of dilatancy. Model is set in a proper thermodynamic framework. However, it should be noted that strictly following thermodynamics for geomaterial behavior, can have negative effects on modeling proper volumetric response (dilatancy and compaction), with alternative material model formulation spaces being suggested by Collins and Houlsby (1997). This is an area which certainly deserves much attention, namely the apparent small disconnect between sound thermodynamic framework for modeling (which nicely applies to metal plasticity) and observed behavior of geomaterials (rock included).

104.8.4.2 Hoek and Brown Model

One of the most often used material models for rock is Hoek and Brown. It is important to note that this model is actually a failure criterion, delineating elastic and failure states of stress, lacking usual plasticity features, such as hardening and/or softening. The most recent edition of the model (Hoek et al., 2002) fixes some earlier observed problems with friction angle determination. This is a valuable model for practical work for rock with low confinement stresses, where behavior is brittle and failure indeed occurs as soon as the failure state of stress is reached. However, in view of shock loading modeling, this model
does not hold much promise, as it lacks, as mentioned above, basic elastic-plastic features.

### 104.8.4.3 Other Models

A number of other material models have been developed for modeling of elastic-plastic-damage behavior of rock. Small selection is presented below. It should be noted that most of those developed models inherit most of their features from models described in previous sections.

Benz and Schwabb (2008) provide comparison for six most commonly used failure criteria for rock. While failure is emphasized, as opposed to full elastic-plastic behavior, the data presented is very telling in view of uncertainty of rock behavior. For example, six deterministic models are calibrated using statistical fitting techniques, and deterministic parameters are developed for a deterministic elastic-plastic (failure) models. The information about the uncertainty of response is thus completely lost.

Das and Basudhar (2009) perform similar exercise with four deterministic models. Moreover, they label some of the test data as outliers thus negatively influencing date regression (removal of statistical moments).

G. W. Ma and Zhou (1998) present an isotropic elastic-plastic model that includes rate dependence, damage development and plasticity of rock. Isotropic damage was used where the elastic constitutive tensor was related to scalar damage parameter $D$

$$E_{ijkl} = E_{ijkl}(1 - D)$$  \hspace{1cm} (104.591)

This is a standard way of incorporating scalar damage (Carol et al., 1995). Of course, anisotropic damage (Rizzi, 1993; Carol et al., 2001a,b; Loret and Rizzi, 1997) is more accurate in modeling realistic materials and should be used whenever possible, however in this case, scalar damage was identified as sufficiently accurate for modeling. In this model, yield strength is controlled by accumulated damage, through a simple linear, isotropic relationship. In addition to that, a non-associated plastic flow is employed. However, plastic flow is limited to deviatoric plane (there is no volumetric component) which reduces accuracy of modeling, since rock material does undergo plastic volumetric change upon plastification (Borja, 2006). Comparison of simulation and experimental data shows somewhat satisfactory similarity, however, bias is present in attenuation plots (similar to results obtained by Wei et al. (2009)).

Bart et al. (2000) presents an interesting approach where an anisotropic poroelastic damage model is used to model slow behavior of rock samples (sandstone). Model is able to predict effects of damage induced by micro-cracking, such as deterioration of elastic and poroelastic properties, induced anisotropy and dilatancy. Model does not feature any plastic deformation, rather inelasticity is completely managed through damage.
Chen et al. (2010) develop an elastic-plastic-damage model which can handle inherent and induced anisotropy. Calibration and application to shale is presented, with main focus on behavior of strongly anisotropic samples. Fabric tensor (not unlike fabric tensor developed for SANISAND family of models described in section 104.6.12) presents an effective modeling tool for modeling anisotropy. However, in their model, Chen and Phoon (2009) simplified modeling of induced anisotropy to isotropic damage (citing complexity of doing it otherwise). This might be unfortunate as induced anisotropy (resulting from anisotropic damage) might be more important than inherent anisotropic, particularly for cases where reversal of loading plays an important role, for example in modeling of shock wave propagation.
104.8.5 Model Calibration / Testing Devices

While a number of material models have been developed over years to model rock behavior, calibration of such models has to be done with great care. Rock is fairly stiff material and as such, stiffness of testing equipment can have significant effects on test results. For example, Labuz and Biolzi (2007) discusses such influence in great detail. Figure 104.8.5 shows how inappropriate stiffness of testing equipment can affect (mask) the real rock response.

![Influence of testing machine stiffness on observed and real test specimen behavior](image)

Figure 104.73: Influence of testing machine stiffness on observed and real test specimen behavior (Labuz and Biolzi, 2007).

In addition to that, elastic-plastic models assume intact rock, so that any influence of discontinuities is removed from test results. This is where scaling of samples plays a very important role. For example Lo et al. (1987) show how variation in test specimen volume (see Figure 104.8.5) can affects (bias) measurements of dynamic elastic modulus, by simply including, within the tested volume, discontinuities and not accounting for them.
Figure 104.74: Influence of volume of test specimen elastic modulus of rock (Lo et al., 1987).
104.8.6 Influence of Pore Fluid Pressure and Temperature on Rock Response

One of the main features of geomaterials is the coupling of pore fluid with the porous solid. Such coupling is taken into account through the effective stress principle. The relationship between effective stress, total stress and pore pressure is (assuming tensile components of stress as positive and compressive pressure, $p$ is positive) (Zienkiewicz et al., 1999a)

$$\sigma''_{ij} = \sigma_{ij} + \alpha \delta_{ij} p$$

(104.592)

where $\sigma''_{ij}$ is effective stress tensor, $\sigma_{ij}$ is total stress tensor, $\delta_{ij}$ is Kronecker delta. $\delta_{ij} = 1$, when $i=j$, and $\delta_{ij} = 0$, when $i \neq j$. For isotropic materials, $\alpha = 1 - K_T/K_S$ (Bouteca and Gueguen, 1999), and $K_T$ is the total bulk modulus of the solid matrix, $K_S$ is the bulk modulus of the solid particle/grains. For most of the geomechanics problems, as the bulk modulus $K_S$ of the solid particles is much larger that that of the whole material, $\alpha \approx 1$ can be assumed. However, in case of rock (as well as for concrete, bone material...), such assumption does not hold all the time so $\alpha$ needs to be kept throughout derivations (Bouteca and Gueguen, 1999).

104.9 Inelastic Behavior and Models for Concrete Beams, Walls and Shells

104.9.1 Uniaxial Material Model for Steel

The uniaxial steel material model used in this study was developed by Menegotto and Pinto (1973) and extended by Filippou et al. (1983). Model is capable of capturing the nonlinear hysteretic behavior and isotropic strain-hardening effect of steel for uniaxial state of stress and strain (1D). The stress–strain response of rebar steel material is shown in Figure 104.90. The model, as presented in Menegotto and Pinto (1973), takes on the form:

$$\sigma^* = b \epsilon^* + \frac{(1 - b) \epsilon^*}{(1 + \epsilon^* R)^1/R}$$

(104.593)

with

$$\epsilon^* = \frac{\epsilon - \epsilon_r}{\epsilon_0 - \epsilon_r}; \quad \sigma^* = \frac{\sigma - \sigma_r}{\sigma_0 - \sigma_r}$$

(104.594)

where $b$ is the strain-hardening ratio, $\epsilon_r$ and $\sigma_r$ are the strain and stress at the point of strain reversal, $\epsilon_0$ and $\sigma_0$ are the strain and stress at the point of intersection of the two asymptotes, $R$ is the curvature parameter that governs the shape of the transition curve between the two asymptotes. It is noted that this model is for uniaxial material behavior, in which the stresses and strains are scalars instead of tensors.
The expression for the curvature parameter $R$ is suggested by Menegotto and Pinto (1973) as:

$$ R = R_0 - \frac{c_{R_1}}{c_{R_2} + \xi} $$

(104.595)

where $R_0$ is the value of the curvature parameter $R$ during initial loading, $c_{R_1}$ and $c_{R_2}$ are degradation parameters that need to be experimentally determined. The parameter $\xi$, which is updated after strain reversal, is defined as:

$$ \xi = \left| \frac{\epsilon_m - \epsilon_0}{\epsilon_y} \right| $$

(104.596)

where $\epsilon_m$ is the maximum (or minimum) strain at the previous strain reversal point, depending on the loading direction of the material. If the current incremental strain is positive, the parameter $\epsilon_m$ takes the value of the maximum reversal strain. Parameter $\epsilon_y$ is the monotonic yield strain.

In order to capture isotropic hardening behavior, Filippou et al. (1983) introduced stress shift mechanism into the original model by Menegotto and Pinto (1973). Note that the hardening rate in compression and tension can be different by choosing different hardening parameters for compression and tension. The proposed relation takes the form:

$$ \frac{\sigma_{st}}{\sigma_y} = a_1 \left( \frac{\epsilon_{max}}{\epsilon_y} - a_2 \right) $$

(104.597)

where $\sigma_{st}$ is the shift stress that determines the shift of yield asymptote, $\epsilon_{max}$ is the absolute maximum strain at strain reversal, and $a_1$ and $a_2$ are hardening parameters in compression which are experimentally determined.
determined. In the case of tension, the hardening parameters $a_1$ and $a_2$ in Equation 104.597 are changed to $a_3$ and $a_4$, that are also determined experimentally or from previous studies for given steel.

104.9.2 3D Plastic Damage Concrete Material Model, Faria-Oliver-Cervera

The concrete material model used in this study was developed by Faria et al. (1998). Model features:

- distinct stress-strain envelopes obtained under compression or under tension
- stiffness recovery after loading reversal
- higher concrete strength under 2D or 3D compression test, compared to 1D loading
- plastic deformations discernible after some compressive stress limit is reached

The material model, as presented in Faria et al. (1998), takes on the form:

$$
\sigma_{ij} = (1 - d^+)\sigma^+_{ij} + (1 - d^-)\sigma^-_{ij}
$$

(104.598)

where, $d^+$ and $d^-$ are scalar damage variables corresponding to tensile and compressive degradation. Cauchy stress tensor $\sigma_{ij}$ involves effective stress components $\sigma^+_{ij}$ and $\sigma^-_{ij}$, that are related to the total effective stress ($\bar{\sigma}_{ij} = \sigma^+_{ij} + \sigma^-_{ij}$), defined as follows:

$$
\bar{\sigma}_{ij} = D_{ijkl}(\varepsilon_{kl} - \varepsilon^p_{kl})
$$

(104.599)

In the previous equation, $D_{ijkl}$ is the fourth order isotropic linear elastic constitutive tensor, $\varepsilon_{kl}$ the small strain tensor and $\varepsilon^p_{kl}$ is the plastic strain tensor. Damage variables together with the plastic strain constitute the internal variable set. Tensile part of the effective stress tensor can be written using principal stresses ($\bar{\sigma}_i$) and principal directions ($p_i$):

$$
\bar{\sigma}^+ = \sum_i \langle \bar{\sigma}_i \rangle p_i \otimes p_i
$$

(104.600)

Compressive components of the effective stress can be written as:

$$
\bar{\sigma}^-_{ij} = \bar{\sigma}_{ij} - \bar{\sigma}^+_{ij}
$$

(104.601)

Following adopted stress split, a tensile equivalent stress $\bar{\tau}^+$ and a compressive equivalent stress $\bar{\tau}^-$ are considered. According to Simo and Ju (1987):

$$
\bar{\tau}^+ = (\bar{\sigma}_{ij} D_{ijkl} \bar{\sigma}^+_{kl})^{1/2}
$$

(104.602)

$$
\bar{\tau}^- = (\sqrt{3}(K\bar{\sigma}_{oct} + \bar{\tau}_{oct}))^{1/2}
$$

(104.603)
In the last equation, $\bar{\sigma}_{\text{oct}}$ and $\bar{\tau}_{\text{oct}}$ are the octahedral normal and shear stress, respectively, obtained from $\bar{\sigma}^-$. $K$ is a material characteristic, adjusted so that 2D and 1D compressive strength ratio can match ratio of 1.16-1.2 (Kupfer et al., 1969). Two separate damage criteria, functions, $g^+$ for tension and $g^-$ for compression, are introduced:

\begin{align}
g^+(\bar{\tau}^+, r^+) &= \tau^+ - r^+ \leq 0 \\
g^-(\bar{\tau}^-, r^-) &= \tau^- - r^- \leq 0
\end{align}

Variables $r^+$ and $r^-$ represent current damage thresholds. Their role is to control the size of expanding damage surfaces. Quadrant $(\bar{\sigma}_2 = 0, \bar{\sigma}_1, \bar{\sigma}_3 \geq 0)$ shows 2D representation for this surface, when $\bar{\tau}^+ = r^+$, Fig 104.76. The bounding surface associated to the principal effective compressive stresses resembles Drucker-Prager cone. It is obvious that the elastic domain under 2D compression is bounded by stresses greater than the 1D elastic compressive stress, denoted by $f_0^-$. 

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure10476.png}
\caption{Initial 2D elastic domain}
\end{figure}

For the plastic flow of the tensor $\dot{\varepsilon}_{ij}^p$, the following is proposed in Faria et al. (1998):

\begin{equation}
\dot{\varepsilon}_{ij}^p = \beta E H(\dot{d}^-) \frac{(\bar{\sigma}_{ij}\dot{\varepsilon}_{ij})}{\sigma_{ij}} D_{ijkl}^{-1} \bar{\sigma}_{kl}
\end{equation}

where $\beta$ is the coefficient which controls the rate of intensity of plastic deformation, $E$ is Young's modulus, $H(\dot{d}^-)$ denotes the Heaviside step function for the compressive damage rate. Proposed model assumes that plastic strain has the direction of the elastic strain tensor $D_{ijkl}^{-1}\bar{\sigma}_{kl}$. It should be noted that the model cannot account for plastic strains for a pure tension test.

Kinematics of damage, internal variables is defined based on the following rate equations:
\[ \dot{d}^+ = \dot{\theta}^+ \frac{\partial G^+(r^+)}{\partial r^+} \quad (104.607) \]
\[ \dot{r}^+ = \dot{\theta}^+ (\geq 0) \quad (104.608) \]

where \( G^+ \) and \( G^- \) are monotonically increasing functions for tension and compression, that are experimentally determined, and \( \dot{\theta}^+ \) and \( \dot{\theta}^- \) are damage consistency parameters. Karush-Kuhn-Tucker conditions implies the following ([Jeremić et al., 1989-2021]):

\[ g^+ \leq 0 \quad \text{damage/plastic function} \quad (104.609) \]
\[ \dot{\theta}^+ \geq 0 \quad \text{consistency parameter} \quad (104.610) \]
\[ g^+ \dot{\theta}^+ = 0 \quad (104.611) \]

which leads to the following conclusions:

- when damage/plasticity function is smaller then zero, \( g^+ < 0 \), consistency parameter must be zero, \( \dot{\theta}^+ = 0 \), implying that no damage occurs,
- when consistency parameter is greater than zero, \( \dot{\theta}^+ > 0 \), damage/plastic function is zero, \( g^+ = 0 \) implying the presence of damage.

It is now possible to define consistency parameter \( \dot{\theta}^+ \) from the damage consistency condition:

\[ g^+(\bar{r}^+, r^+) = 0 \implies \dot{r}^+ = \dot{\theta}^+, \quad \dot{\theta}^+ \geq 0 \quad (104.612) \]

Introducing equation \((104.612)\) into \((104.607)\) the flow rule for the tensile damage variable can be expressed as:

\[ \dot{d}^+ = \frac{\partial G^+(r^+)}{\partial r^+} \dot{r}^+ = \dot{G}^+(r^+) \geq 0 \quad (104.613) \]

The compression damage variable is then:

\[ \dot{d}^- = \frac{\partial G^-(r^-)}{\partial r^-} \dot{r}^- = \dot{G}^-(r^-) \geq 0 \quad (104.614) \]
104.10 Calibration of Elastic-Plastic Material Models

104.10.1 Calibration of Elastic-Plastic Material Models, Soil

104.10.2 Calibration of Elastic-Plastic Material Models, Rock

104.10.3 Calibration of Elastic-Plastic Material Models, Contact/Joint/Interface

104.10.4 Calibration of Elastic-Plastic Material Models, Concrete

104.10.5 Calibration of Elastic-Plastic Material Models, Steel

104.11 Energy Dissipation Calculations for Solids

This section is based on Yang et al. (2018)

104.11.1 Introduction

Energy dissipation in elastic plastic solids and structures is the result of an irreversible dissipative process in which energy is transformed from one form to another and entropy is produced. The transformation and dissipation of energy is related to permanent deformation and damage within an elastic-plastic material. Of particular interest here is the dissipation of mechanical energy that is input into elastic-plastic solids by static or dynamic excitations.

Early work on plastic dissipation was done by Farren and Taylor (1925) and Taylor and Quinney (1934). They performed experiments on metals and proved that a large part, but not all, of the input mechanical energy is converted into heat. The remaining part of the non-recoverable plastic work is known as the stored energy of cold work. The ratio of plastic work converted into heating (Quinney–Taylor coefficient), usually denoted as $\beta$, has been used in most later work on this topic. Based on large amount of experimental data, this ratio was determined to be a constant between 0.6 to 1.0 (Clifton et al., 1984; Belytschko et al., 1991; Zhou et al., 1996; Dolinski et al., 2010; Ren and Li, 2010; Osovski et al., 2013).

More recently Rittel (Rittel, 2000; Rittel and Rabin, 2000; Rittel et al., 2003) published several insightful papers on the energy dissipation (heat generation) of polymers during cyclic loading, presenting both experimental and theoretical works. Rosakis et al. Rosakis et al. (2000) presented a constitutive model for metals based on thermoplasticity that is able to calculate the evolution of energy dissipation. Follow up papers (Hodowany et al., 2000; Ravichandran et al., 2002) present assumptions to simplify the problem. One direct application of plastic dissipation to geotechnical engineering is presented by Veveakis et al. (Veveakis et al., 2007, 2012), using thermoporomechanics to model the heating and pore...
pressure increase in large landslides, like the 1963 Vajont slide in Italy.

In the past few decades, extensive studies have been conducted on energy dissipation in structures and foundations. Work by Uang and Bertero (1990) has been considered a source and a reference for many recent publications dealing with energy as a measure of structural demand. Uang and Bertero (1990) developed an energy analysis methodology based on absolute input energy (or energy demand). Numerical analysis results were compared with experiments on a multi-story building. In work by Uang and Bertero (1990), hysteretic energy is calculated indirectly by taking the difference of absorbed energy and elastic strain energy. The term absorbed energy of each time step is simply defined as restoring force times incremental displacement. It is also stated that hysteretic energy is irrecoverable, which indicates that this parameter was considered the same as hysteretic dissipation or plastic dissipation. An equation for energy balance, is given by (Uang and Bertero (1990)) as:

\[ E_i = E_k + E_\xi + E_a = E_k + E_\xi + E_s + E_h \]

where \( E_i \) is the (absolute) input energy, \( E_k \) is the (absolute) kinetic energy, \( E_\xi \) is the viscous damping energy, \( E_a \) is the absorbed energy, which is composed of elastic strain energy \( E_s \) and hysteretic energy \( E_h \).

The problem with this approach is the absence of plastic free energy, which is necessary to correctly evaluate energy dissipation of elastic-plastic materials and to uphold the second law of thermodynamics. While there is no direct plot of plastic dissipation (hysteretic energy) in Uang and Bertero (1990), since it was not defined directly, there are plots of other energy components. Plastic dissipation can be easily calculated from these plots. After doing this, indications of negative incremental energy dissipation, which violates the basic principles of thermodynamics, were found in various sections of the paper.

This misconception could be clarified by renaming hysteretic energy as plastic work, a sum of plastic dissipation and plastic free energy. Both plastic work and plastic free energy can be incrementally negative, but plastic dissipation (defined as the difference of plastic work and plastic free energy) must be incrementally non-negative during any time period. Unfortunately, this misconception has been inherited (if not magnified) by many following studies on energy analysis of earthquake soils and structures (hundreds of papers).

The basic principles of thermodynamics are frequently used to derive new constitutive models, for example by Dafalias and Popov (1975), Ziegler and Wehrli (1987), Collins and Houlbysy (1997), Houlsby and Puzrin (2000), Collins (2002), Collins and Kelly (2002), Collins (2003) and Feigenbaum and Dafalias (2007). The concept of plastic free energy is introduced to enforce the second law of thermodynamics for developed constitutive models. It is important to distinguish between energy dissipation due to plasticity and plastic work, which is often a source of a confusion. Plastic work is the combination of plastic free
energy and plastic energy dissipation, which is defined as the amount of heat (and other forms of energy) transformed from mechanical energy during an irreversible dissipative process. The physical nature of plastic free energy is illustrated later in this section through a conceptual example that is analyzed on particle scale. Essentially, development of plastic free energy is caused by particle rearrangement in granular assembly under external loading.

Specific formulation of free energy depends on whether the elastic and plastic behavior of the material is coupled. According to Collins et al. Collins and Houlsby (1997), Collins (2002), Collins (2003), material coupling behavior can be divided into modulus coupling, where the instantaneous elastic stiffness (or compliance) moduli depend on the plastic strain, and dissipative coupling, where the rate of dissipation function depends not only on the plastic strains and their rates of change but also on the stresses (or equivalently the elastic strains). The modulus coupling describes the degradation of stiffness as in for rock and concrete, and is usually modeled by employing a coupled elastic-plastic constitutive model or by introducing damage variables. The dissipative coupling is considered to be one of the main reasons for non-associative behavior in geomaterials Collins and Houlsby (1997), Ziegler (1981).

A number of stability postulates are commonly used to prevent violation of principles of thermodynamics. Stability postulates include Drucker’s stability condition Drucker (1956), Drucker (1957), Hill’s stability condition Bishop and Hill (1951), Hill (1958), and Il’Iushin’s stability postulate Il’Iushin (1961), Lubliner (1990). As summarized in a paper by Lade Lade (2002), theoretical considerations by Nemat–Nasser (1983) and Runesson and Mróz (1989) have suggested that they are sufficient but not necessary conditions for stability. These stability postulates can indeed ensure the admissibility of the constitutive models by assuming certain restrictions on incremental plastic work. As demonstrated by Collins (2002), if the plastic strain rate is replaced by the irreversible stain rate in Drucker’s postulate, then all the standard interpretations of the classical theory still apply for coupled materials. Dafalias (1977) also modified Il’Iushin’s postulate in a similar way and applied it to both coupled and uncoupled materials.

It is important to note that development of inelastic deformation in geomaterials involves large changes in entropy, and significant energy dissipation. It is thus useful to perform energy dissipation (balance) analysis for all models with inelastic deformation. In section we focus on energy dissipation on material level. Focus is on proper modeling that follows thermodynamics. Comparison is made between accumulated plastic dissipation and accumulated plastic work, since these quantities can be quite different in most cases. As a way of verification, the input work, which is introduced by applying external forces, is compared with the stored energy and dissipation in the entire system. Finally, conclusions on plastic energy dissipation are drawn from the verified results.
104.11.2 Theoretical and Computational Formulations

104.11.2.1 Thermo-Mechanical Theory

For energy analysis of elastic-plastic materials undergoing isothermal process, it is beneficial to start from the statement of the first and second laws of thermodynamics:

$$\dot{W} = \dot{\Psi} + \Phi$$ (104.616)

where $\Phi \geq 0$ and $\dot{W} \equiv \sigma : \dot{\varepsilon} = \sigma_{ij} \dot{\varepsilon}_{ij}$ is the rate of work per unit volume. The function $\Psi$ is the Helmholtz free energy, and $\Phi$ is the rate of dissipation; both defined per volume. The free energy $\Psi$ is a function of the state variables (also known as internal variables), but $\Phi$ and $\dot{W}$ are not the time derivatives of the state functions. The choice of state variables depends on the complexity of constitutive model that is being used, as cyclic loading with certain hardening behaviors usually requires more state variables. This will be elaborated in the following sections as we discuss specific elastic-plastic material models.

For general elastic-plastic materials, the free energy depends on both the elastic and plastic strains. In most material models, it can be assumed that the free energy $\Psi$ can be decomposed into elastic and plastic parts:

$$\Psi = \Psi_{el} + \Psi_{pl}$$ (104.617)

The total rate of work associated with the effective stress can be written as the sum of an elastic and plastic component:

$$\dot{W}^{el} \equiv \sigma_{ij} \dot{\varepsilon}_{el}^{ij} = \dot{\Psi}_{el}$$ (104.618)

and

$$\dot{W}^{pl} \equiv \sigma_{ij} \dot{\varepsilon}_{pl}^{ij} = \dot{\Psi}_{pl} + \Phi$$ (104.619)

Note that the focus of this section is the energy dissipation caused by material plasticity, which should be distinguished from viscous coupling and other sources of energy dissipation. So the effects of solid-fluid interaction are neglected and all stresses are defined as effective stresses in further derivations. In order to avoid confusion, the common notation ($\sigma'_{ij}$) will not be used. Standard definition of stress from mechanics of materials, i.e. positive in tension, is used.

In the case of a decoupled material, the elastic free energy $\Psi_{el}$ depends only on the elastic strains, and the plastic free energy $\Psi_{pl}$ depends only on the plastic strains, as shown by Collins and Houlsby (1997):

$$\Psi = \Psi_{el}(\varepsilon_{el}^{ij}) + \Psi_{pl}(\varepsilon_{pl}^{ij})$$ (104.620)
The effective stress can also be decomposed into two parts:

\[ \sigma_{ij} = \alpha_{ij} + \chi_{ij} \]  \hspace{1cm} (104.621)

where \( \chi_{ij} \) is a stress-like variable that is related to the dissipative behavior of elastic-plastic material. The difference between the actual stress \( \sigma_{ij} \) and the stress-like variable \( \chi_{ij} \) is another stress-like term \( \alpha_{ij} \), which is defined from the plastic free energy function \( \Psi_{pl} \). In simple kinematic hardening models, this variable \( \alpha_{ij} \) controls the shift behavior of stress under cyclic loading, and thus usually referred to as shift or back stress.

Ziegler’s orthogonal postulate Ziegler and Wehrli (1987) ensures the validity of Equation 104.621. It is equivalent to the maximum entropy production criterion, which is necessary to obtain unique formulation. Also, this is a weak assumption so that all the major continuum models of thermo-mechanics are included. Equation 104.619 of plastic work rate can hence be rewritten as:

\[ \dot{W}_{pl} \equiv \sigma_{ij} \dot{\varepsilon}_{ij} = \dot{\Psi}_{pl} + \Phi = \alpha_{ij} \dot{\varepsilon}_{ij} + \chi_{ij} \dot{\varepsilon}_{ij} \]  \hspace{1cm} (104.622)

The plastic work \( \dot{W}_{pl} \) is the product of the actual Cauchy stress \( \sigma_{ij} \) with the plastic strain rate \( \dot{\varepsilon}_{ij} \), while the dissipation rate \( \Phi \) is the product of the stress variable \( \chi_{ij} \) with the plastic strain rate \( \dot{\varepsilon}_{ij} \). They are only equal if the rate of plastic free energy \( \dot{\Psi}_{pl} \) is zero, or equivalently, if the free energy depends only on the elastic strains.

In kinematic hardening models, where the back stress describes the translation (or rotation) of the yield surface, the decomposition of the true stress (sum of back stress and dissipative stress) is a default assumption. Although such a shift stress is important for anisotropic material models, Collins and Kelly (2002) have pointed out that it is also necessary in isotropic models of geomaterials with different strength in tension and compression.

### 104.11.2.2 Plastic Free Energy

A popular conceptual model, which focused on particulate materials and demonstrated the physical occurrence of shift stresses, was described by Besseling and Van Der Giessen (1994) and Collins and Kelly (2002). On macro (continuum) scale, every point in a given element is at yield state and deforms plastically. But on meso-scale, only part of this element is undergoing plastic deformations, the remaining part is still within yield surface and respond elastically. The elastic strain energy stored in the elastic part of a plastically deformed macro-continuum element is considered to be locked into the macro-deformation, giving rise to the plastic free energy function \( \Psi_{pl} \) and its associated back stress \( \alpha_{ij} \). This energy can be released only when the plastic strains are reversed.
For better explanation, the nature of plastic free energy in particulate materials is illustrated through a finite element simulation combined with considerations of particle rearrangement on mesoscopic scale. Figure 104.77 shows stress-strain response of Drucker-Prager with nonlinear Armstrong-Frederick kinematic hardening, a typical elastic-plastic model for metals and geomaterials. Six states during shear are chosen to represent evolution of micro fabric of the numerical sample. Correspondingly, Figure 104.78 shows the process of particle rearrangement of the 2D granular assembly under cyclic shearing from microscopic level. The square window can be roughly considered as a representative volume (a constitutive level or a finite element) in FEM.

![Stress-strain and stress-plastic strain versus time](image)

**Figure 104.77:** Elastic-plastic material modeled with Drucker-Prager yield function and Armstrong-Frederick kinematic hardening under cyclic shear loading: (a) Stress-strain curve; (b) stress and plastic strain versus time.

By discussing movement and energy of particle A in Figure 104.78, the physical nature of plastic free energy is illustrated. At state (a), which is the beginning of deformation, particle A does not bear any load other than its self weight. State (b) is in middle of loading, when particle B pushes downwards to particle A until it makes contact with particle D and E. Load reaches peak at state (c), and there’s no space for particle A to move. Then the sample is unloaded to state (d). Particle A is now stuck between particles C, D, and F, which means that certain amount of elastic energy is stored due to particle elastic deformation. Compared with state (a), this part of elastic energy is not released when the sample is unloaded, which indicates that it’s not classic strain energy. This part of elastic energy on particle level which can’t be released by unloading is defined as the plastic free energy in granular materials. Reverse loading starts at state (e), where particle D pushes particle A upwards, making it squeeze through particle C and F. Elastic energy on particle level, which is now defined as plastic free energy, is released during...
Figure 104.78: Particle rearrangement of a 2D granular assembly under cyclic shearing: (a) Initial state; (b) Loading (accumulating plastic free energy); (c) End of loading (maximum plastic free energy); (d) Unloading (plastic free energy unchanged); (e) Reverse loading (releasing plastic free energy); (f) End of reverse loading (plastic free energy released).
reverse loading.

By analyzing this example, an explanation on particle scale is provided for the origin of plastic free energy in granular materials. It is important to note that the concept of plastic free energy also exists in metals and other materials, as studied by Dafalias et al. (2002) and Feigenbaum and Dafalias (2007). The physical nature of plastic free energy in these materials can be different and probably should be studied on molecular and/or crystalline level.

Collins (Collins and Kelly 2002, Collins 2003) suggested that in the case of granular materials, the particle-level plastic energy dissipation during normal compaction, arises from the plastic deformations occurring at the inter-granular contacts on the strong force chains, that are bearing the bulk of the applied loads. Collins also suggested that the locked-in elastic energy is produced in the weak force networks, where the local stresses are not large enough to produce plastic deformation at the grain contacts. The plastic strains can be associated with the irreversible rearrangement of the particles, whilst the elastic energy arises from the elastic compression of the particle contacts. Part of this elastic strain energy will be released during unloading, however other part of this energy will be trapped as a result of the irreversible changes in the particle configuration.

104.11.2.3 Plastic Dissipation

As pointed out, plastic work and energy dissipation are not the same physical quantity. The confusion of these two concepts often leads to incorrect results and conclusions, especially in seismic energy dissipation analysis. Of major concern in this section is the computation of plastic dissipation, as elaborated in this section.

With the decoupling assumption (Equation 104.620), the second law of thermodynamics (positive entropy production) directly leads to the dissipation inequality, which states that the energy dissipated due to the difference of the plastic work rate and the rate of the plastic part of the free energy must be non-negative:

$$\Phi = \sigma_{ij} \dot{\epsilon}_{ij}^{pl} - \dot{\Psi}_{pl} = \sigma_{ij} \dot{\epsilon}_{ij}^{pl} - \rho \dot{\psi}_{pl} \geq 0$$

(104.623)

where $\dot{\psi}_{pl}$ is the rate of plastic free energy, per unit mass, and $\rho$ is the mass density. In addition, $\psi_{pl}$ denotes plastic free energy density, which is generally not constant at different locations in a body. This expression is closer to physics and makes it convenient for further derivations.

Now we proceed to consider how to calculate plastic free energy, which can then be used to calculate dissipation. According to Feigenbaum and Dafalias (2007), plastic free energy density $\psi_{pl}$ is assumed to be additively decomposed into parts which correspond to the isotropic, kinematic and distortional
hardening mechanisms as follows:

\[
\psi_{pl} = \psi_{pl}^{iso} + \psi_{pl}^{ani} \quad \psi_{pl}^{ani} = \psi_{pl}^{kin} - \psi_{pl}^{dis}
\]

(104.624)

where \(\psi_{pl}^{iso}\), \(\psi_{pl}^{ani}\), \(\psi_{pl}^{kin}\), and \(\psi_{pl}^{dis}\) are the isotropic, anisotropic, kinematic, and distortional parts of the plastic free energy, respectively. The anisotropic part is assumed to decompose into kinematic and distortional parts, which correspond to different hardening models. The subtraction, instead of addition, of \(\psi_{pl}^{dis}\) from \(\psi_{pl}^{kin}\), to obtain the overall anisotropic part \(\psi_{pl}^{ani}\) of the plastic free energy, is a new concept proposed by Feigenbaum and Dafalias (2007). This expression can better fit experimental data, as well as satisfy the plausible expectations for a limitation of anisotropy development. The distortional part of the plastic free energy \(\psi_{pl}^{dis}\) is related to the directional distortion of yield surface and will only be present if the material model incorporates distortional strain hardening, which is not considered in the formulations and examples of this study.

As pointed out by Dafalias et al. (2002), the thermodynamic conjugates to each of the internal variables exist and each part of the plastic free energy can be assumed to be only a function of these conjugates. The explicit expressions for the isotropic and kinematic components of the plastic free energy are:

\[
\psi_{pl}^{iso} = \psi_{pl}^{iso}(\bar{k}) = \frac{\kappa_1}{2\rho} \bar{k}^2; \quad \psi_{pl}^{kin} = \psi_{pl}^{kin}(\bar{\alpha}_{ij}) = \frac{a_1}{2\rho} \bar{\alpha}_{ij} \bar{\alpha}_{ij}
\]

(104.625)

where \(\bar{k}\) and \(\bar{\alpha}_{ij}\) are the thermodynamic conjugates to \(k\) (size of the yield surface) and \(\alpha_{ij}\) (deviatoric back stress tensor representing the center of the yield surface), respectively. Material constants \(\kappa_1\) and \(a_1\) are non-negative material constants whose values depend on the choice of elastic-plastic material models.

According to definition, the thermodynamic conjugates are related to the corresponding internal variables by:

\[
k = \rho \frac{\partial \psi_{pl}^{iso}}{\partial \bar{k}} = \kappa_1 \bar{k}; \quad \alpha_{ij} = \rho \frac{\partial \psi_{pl}^{kin}}{\partial \bar{\alpha}_{ij}} = a_1 \bar{\alpha}_{ij}
\]

(104.626)

By substituting Equation 104.626 back into Equation 104.625, the plastic free energy can be expressed in terms of the internal variables:

\[
\psi_{pl}^{iso} = \frac{1}{2\rho \kappa_1} \bar{k}^2; \quad \psi_{pl}^{kin} = \frac{1}{2\rho a_1} \bar{\alpha}_{ij} \bar{\alpha}_{ij}
\]

(104.627)

With Equation 104.627, the components of plastic free energy can be computed, as long as the internal variables are provided. Combining Equation 104.623 with 104.627, the plastic dissipation in a given elastic-plastic material can be accurately obtained at any location, at any time. This approach allows engineers and designers to correctly identify energy dissipation in time and space and make appropriate conclusions on material behavior.
104.11.2.4 Energy Computation in Finite Elements

Formulations from the previous section are applied to FEM analysis in order to follow energy dissipation. Energy density is chosen as the physical parameter for energy analysis. Energy density in this study is defined as the amount of energy stored in a given region of space per unit volume.

For FEM simulations, both external forces and displacements can be prescribed. The finite element program accepts either (or both) forces and/or displacements as input and solves for the other. Either way, the rate of input work can be calculated by simply multiplying force and displacement within a time step. Therefore, input work of a finite element model is:

\[ W_{\text{Input}}(t) = \int_0^t \dot{W}_{\text{Input}}(T) dT = \int_0^t \sum_i F_{i \text{ex}}(x, T) \dot{u}_i(x, T) dT \]  

(104.628)

where \( F_{i \text{ex}} \) is the external force and \( u_i \) is the displacement computed at the location of the applied load, at given time step, for a load controlled analysis. The external load can have many forms, including nodal loads, surface loads, and body loads. All of them are ultimately transformed into nodal forces. As shown in Equation 104.628, input work is computed incrementally at each time step, in order to obtain the evolution of total input work at certain time.

As shown in Figure 104.79, when loads and/or displacements are introduced into a finite element model, the input energy will be converted in a number of different forms as it propagates through the system. Input energy will be converted into kinetic energy, free energy, and dissipation. As mentioned before, free energy can be further separated into elastic part, which is traditionally defined as strain energy, and plastic part, which is defined as the plastic free energy. Kinetic energy and strain energy can be considered as the recoverable portion of the total energy since they are transforming from one to another. Plastic free energy is more complicated in the sense that it is conditionally recoverable during reverse loading, as has been discussed in detail in previous sections. Other than kinetic energy and free energy, the rest of the input energy is dissipated, transformed into heat or other forms of energy that are irrecoverable.

Calculation of kinetic energy and strain energy is rather straightforward:

\[ U_K(x, t) = \frac{1}{2} \rho \dot{u}_{ij}(x, t) \dot{u}_{ij}(x, t) \]  

(104.629)

\[ U_S(x, t) = \int_0^t \dot{U}_S(x, T) dT = \int_0^t \sigma_{ij}(x, T) \dot{\varepsilon}_{ij}^e(x, T) dT \]  

(104.630)

where \( U_K \) and \( U_S \) are the kinetic energy density and strain energy density, respectively.
Similar to the input energy, strain energy density and plastic free energy are also computed incrementally. Integrating energy density over the entire model, corresponding energy quantities are expressed as:

\[
E_K(t) = \int_V U_K(x, t)dV
\]  \hspace{1cm} (104.631)

\[
E_S(t) = \int_V U_S(x, t)dV
\]  \hspace{1cm} (104.632)

\[
E_P(t) = \int_V \Psi_{pl}(x, t)dV
\]  \hspace{1cm} (104.633)

where \(E_K\), \(E_S\), and \(E_P\) are the kinetic energy, strain energy, and plastic free energy of the entire model, respectively. Energy densities, defined in Equations 104.629 and 104.630 are functions of both time and space, while energy components, defined in the above equations (Equation 104.631, 104.632, and 104.633) are only functions of time, since they are integrated over the whole model.

Although the plastic free energy is conditionally recoverable, it is still considered to be stored in the system, rather than dissipated. Summing up all the stored energy \(E_{\text{stored}}\), one obtains:

\[
E_{\text{stored}} = E_K + E_S + E_P
\]  \hspace{1cm} (104.634)
Rate of plastic dissipation, given by Equation 104.623, can be integrated over time and space:

\[ D_P(t) = \int_V \int_0^t \Phi(x, T) \,dT \,dV \]  

(104.635)

where \( D_P \) is the dissipation due to plasticity of the entire model at certain time.

Finally the energy balance of a finite element model is given by:

\[ W_{Input} = E_{Stored} + D_P = E_K + E_S + E_P + D_P \]  

(104.636)

### 104.11.3 Numerical Studies

Numerical simulation results presented in this section are performed using the Real-ESSI (Jeremić et al., 1988-2021). Examples in this section focus on constitutive behavior of elastic-plastic material from the perspective of energy dissipation.

All cases are assumed to be static problems. External loads are applied incrementally using load- or displacement-control scheme. System equations are solved using Newton-Raphson iteration algorithm and UMFPACK solver. Standard 8-node-brick elements are used in all cases, in order to eliminate the variation in energy computation caused by different element types.

#### 104.11.3.1 Elastic Material

Initial investigation of energy dissipation is focused on linear elastic material. It is noted that linear elastic material does not dissipate energy. Use of linear elastic material model is suitable for preliminary verification of the newly developed energy analysis methodology. In this section, energy balance in a single brick element and a cantilever beam is studied, as shown in Figure 104.80.

It should be mentioned that the bending deformations of cantilever are not accurate due to the use of a single layer of 8-node-brick elements in the direction of stress and strain variation. However, the focus
of this example is energy transformation and balance, which are not affected by inaccurate deformations in this example.

The simplest case is a single element model under uniform shear load. The model is constrained appropriately to simulate simple shear test. In order to show the influence of different material parameters and loads, a set of simulations are performed and the results are presented in Table 104.2 and Figure 104.81.

Table 104.2: Energy analysis results for linear elastic materials (single element).

<table>
<thead>
<tr>
<th>Material Property</th>
<th>Simulation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$ (GPa) $\nu$ $u$ (m) $W_{Input}$ (J)</td>
<td>$E_K$ (J) $E_S$ (J) $E_P$ (J) $E_{Stored}$ (J) $D_P$ (J)</td>
</tr>
<tr>
<td>100 0.30 2.60E-5 13.00</td>
<td>0.00 13.00 0.00 13.00 0.00</td>
</tr>
<tr>
<td>150 0.30 1.73E-5 8.67</td>
<td>0.00 8.67 0.00 8.67 0.00</td>
</tr>
<tr>
<td>200 0.30 1.30E-5 6.50</td>
<td>0.00 6.50 0.00 6.50 0.00</td>
</tr>
<tr>
<td>250 0.30 1.04E-5 5.20</td>
<td>0.00 5.20 0.00 5.20 0.00</td>
</tr>
<tr>
<td>300 0.30 8.67E-6 4.33</td>
<td>0.00 4.33 0.00 4.33 0.00</td>
</tr>
<tr>
<td>200 0.20 1.20E-5 6.00</td>
<td>0.00 6.00 0.00 6.00 0.00</td>
</tr>
<tr>
<td>200 0.25 1.25E-5 6.25</td>
<td>0.00 6.25 0.00 6.25 0.00</td>
</tr>
<tr>
<td>200 0.30 1.30E-5 6.50</td>
<td>0.00 6.50 0.00 6.50 0.00</td>
</tr>
<tr>
<td>200 0.35 1.35E-5 6.75</td>
<td>0.00 6.75 0.00 6.75 0.00</td>
</tr>
<tr>
<td>200 0.40 1.40E-5 7.00</td>
<td>0.00 7.00 0.00 7.00 0.00</td>
</tr>
</tbody>
</table>

Since linear elastic material is used with static algorithm, energy components related to dynamics (kinetic energy) and plasticity (plastic free energy and plastic dissipation) are equal to zero. This means that all input work is stored in the system, as observed in all cases.

Figure 104.81 shows that energy stored in the system is inversely proportional to Young’s moduli $E$ and proportional to one plus Poisson’s ratio $(1 + \nu)$. This is expected because of the following equations for strain energy under static shear loading:

$$E_S = \frac{1}{2} \tau \gamma = \frac{1}{2G} \tau^2 = \frac{1 + \nu}{E} \tau^2$$  \hspace{1cm} (104.637)

Note that these relationships are only valid at constitutive level. For models with more finite elements, stress and strain are generally not uniform. The computation of energy depends on the distribution of energy density, and nonuniform stress/strain distribution will result in nonuniform energy density distribution.
In order to study the influence of simulation parameters in larger models, another set of simulations with cantilever model (Figure 104.80b) are performed. Vertical loads are applied to the nodes of the free end. In this case, both shearing and bending occurs, which means that in general a full 3D state of stress and strain is present. The results are presented in Table 104.3 and Figure 104.81. As expected, energy behavior of cantilever is different than the single-element/constitutive example.

For all cases, the energy balance between input and storage is maintained, which gives us confidence on the energy calculation methodology for elastic material. According to results in Figure 104.81, energy stored in the system is still inversely proportional to Young’s modulus. This is because the general equation for elastic strain energy density is:

\[
E_s = \frac{1}{2E} \left( \sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 + 2(1 + \nu)(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2) \right)
\]  

(104.638)

So as long as all the elements have the same Young’s modulus, the relationship between stored energy and Young’s modulus will remain valid.

104.11.3.2 von Mises Plasticity

Elastic-plastic modeling using von Mises material model has been proven to be effective in modeling pressure-independent materials like steel or other metals. In this section, the energy behavior of models using von Mises plasticity with various hardening rules are examined using the proposed method. The material model parameters used in this section are summarized in Table 104.4.

Note that associated plasticity is used in all models in this section, which means that the plastic flow
Table 104.3: Energy analysis results for linear elastic materials (cantilever model).

<table>
<thead>
<tr>
<th>Material Property</th>
<th>Simulation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$ (GPa)</td>
<td>$\nu$</td>
</tr>
<tr>
<td>100</td>
<td>0.30</td>
</tr>
<tr>
<td>150</td>
<td>0.30</td>
</tr>
<tr>
<td>200</td>
<td>0.30</td>
</tr>
<tr>
<td>250</td>
<td>0.30</td>
</tr>
<tr>
<td>300</td>
<td>0.30</td>
</tr>
<tr>
<td>200</td>
<td>0.20</td>
</tr>
<tr>
<td>200</td>
<td>0.25</td>
</tr>
<tr>
<td>200</td>
<td>0.30</td>
</tr>
<tr>
<td>200</td>
<td>0.35</td>
</tr>
<tr>
<td>200</td>
<td>0.40</td>
</tr>
</tbody>
</table>

direction $m_{ij}$ is equal to the gradient of the yield surface $n_{ij} (= \partial f/\partial \sigma_{ij})$. Since the yield function is of von Mises type, associated plasticity leads to the result that only deviatoric plastic flow will appear in all cases.

**No Hardening (Elastic-Perfectly Plastic).** In this example, elastic-perfectly plastic material is used. Equations 104.623 and 104.627 indicate that in the case of no hardening the rate of plastic free energy is zero. Then the incremental plastic work is equal to incremental plastic dissipation. Note that this is one of the rare cases where plastic dissipation equals to plastic work.

Figure 104.82 shows stress–strain curve (left) and energy calculated for elastic-perfectly plastic constitutive model (right) used here.

In this case, the plastic dissipation is equal to the plastic work. This means that the plastic free energy does not develop at all during loading and unloading. Zero plastic free energy points out the absence of fabric evolution of a particulate, elastic-plastic material, as all the input work is dissipated through particle to particle friction. Since there is no plastic free energy $E_P$ in this case, the stored energy equals to mechanical energy, which is the combination of strain energy $E_S$ and kinetic energy $E_K$. Total stored energy $E_{Stored}$ develops nonlinearly and always has the same value at the beginning of every loop after the first one. Plastic dissipation $D_P$ increases linearly when the material yields. This
Table 104.4: Model parameters for cases using von Mises plasticity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>No Hardening</th>
<th>Linear Isotropic</th>
<th>Linear Kinematic</th>
<th>A–F Kinematic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mass_density</td>
<td>kg/m³</td>
<td>8050</td>
<td>8050</td>
<td>8050</td>
<td>8050</td>
</tr>
<tr>
<td>elastic_modulus</td>
<td>GPa</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>poisson_ratio</td>
<td></td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>von_mises_radius</td>
<td>MPa</td>
<td>250</td>
<td>250</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>isotropic_hardening_rate</td>
<td>GPa</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kinematic_hardening_rate</td>
<td>GPa</td>
<td>0</td>
<td>50</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>armstrong_frederick_ha</td>
<td>GPa</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>armstrong_frederick_cr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 104.82: Energy analysis of elastic-plastic material modeled using von Mises plasticity with no hardening: (a) Stress–strain curve; (b) Input work, plastic dissipation, strain energy and plastic work.

can be explained by rewriting Equation 104.623 with \( \Psi_{pl} = 0 \):

\[
\Phi = \sigma_{ij} \dot{\epsilon}_{ij}^{pl}
\]  

(104.639)

where stress \( \sigma_{ij} \) is constant after elastic perfectly plastic material yields, and rate plastic deformation \( \dot{\epsilon}_{ij}^{pl} \) is also constant. Then the rate of plastic dissipation is constant which makes the plastic dissipation \( D_P \) increase linearly.
Linear Isotropic Hardening. Next material model used is von Mises plasticity with linear isotropic hardening. First used to model monotonic behavior of elastic-plastic materials, isotropic hardening assumes that the yield surface maintains shape, while isotropically (proportionally) changing its size. Figure 104.83 illustrates the stress–strain response as well as energy balance for elastic-plastic material with isotropic hardening.

![Stress–strain curve and energy analysis](image)

Figure 104.83: Energy analysis of elastic-plastic material modeled using von Mises plasticity with linear isotropic hardening: (a) Stress–strain curve; (b) Input work, plastic dissipation, strain energy, and plastic work.

As can be observed from Figure 104.83, plastic free energy is equal to the plastic work, which means that the plastic dissipation is zero during cycles of loading. Even though this might sound surprising, it can be explained using basic thermodynamics. Linear isotropic hardening, used in this case, can be described through a rate of the internal variable (size of the yield surface) \( \dot{k} \) as:

\[
\dot{k} = k_1 |\dot{e}_{ij}^{pl}|
\]  

(104.640)

where \( |\dot{e}_{ij}^{pl}| \) is the magnitude of the rate of plastic strain while \( k_1 \) is a hardening constant. The hardening constant \( k_1 \) is denoted as isotropic hardening rate in Table 104.4. Substituting previous equation into Equation 104.627 yields:

\[
\psi_{pl} = \psi_{iso} = \frac{k_1}{2\rho} \dot{e}_{ij}^{pl} e_{ij}^{pl}
\]

(104.641)

Take the time derivative of the above equation:

\[
\dot{\psi}_{pl} = \frac{k_1}{\rho} e_{ij}^{pl} e_{ij}^{pl}
\]

(104.642)
Then the rate of dissipation due to plasticity can be expressed as:

\[
\Phi = \sigma_{ij} \dot{\epsilon}_{ij}^{pl} - \rho \omega_{pl} = (\sigma_{ij} - \kappa_1 \epsilon_{ij}^{pl}) \dot{\epsilon}_{ij}^{pl} = (\sigma_{ij} - km_{ij}) \dot{\epsilon}_{ij}^{pl}
\]  

(104.643)

where \(m_{ij}\) is the plastic flow direction. The plastic flow direction defines the direction of incremental plastic strain, which can be different from the direction of total plastic strain. But in the case of associated von Mises plasticity with only isotropic hardening, the plastic flow direction \(m_{ij}\) is the same as the direction of the total plastic strain \(\epsilon_{ij}^{pl}\). Thus we have \(\kappa_1 \epsilon_{ij}^{pl} = km_{ij}\) in the above equation.

Substitute the plastic flow direction \(m_{ij}\) with the gradient of yield surface \(n_{ij}\), and also note that \(\sigma_{ij} \dot{\epsilon}_{ij}^{pl} = s_{ij} \dot{\epsilon}_{ij}^{pl}\), where \(s_{ij} = \sigma_{ij} - 1/3 \delta_{ij} \sigma_{kk}\) is the deviatoric part of the stress tensor, the rate of plastic dissipation can be rewritten as:

\[
\Phi = (s_{ij} - km_{ij}) \dot{\epsilon}_{ij}^{pl} = \alpha_{ij} \dot{\epsilon}_{ij}^{pl}
\]  

(104.644)

Realizing that the back stress \(\alpha_{ij}\) is always zero since we assume no kinematic hardening, then the rate of plastic dissipation becomes zero, which means there is no energy dissipation during cycles of loading for isotropically hardening material. Obviously, the observed response is not physical from the perspective of energy dissipation. Therefore, isotropic hardening material models cannot properly model energy dissipation, even for monotonic loading.

**Prager Linear Kinematic Hardening.** Compared with isotropic hardening, kinematic hardening can better describe the constitutive, stress-strain behavior of elastic-plastic materials, particularly for cyclic loading. Elastic-plastic material that relies on kinematic hardening is used to analyze energy dissipation. Both linear and nonlinear kinematic hardening rules are investigated in relation to energy dissipation.

Prager’s linear kinematic hardening rule is given as:

\[
\dot{\alpha}_{ij} = a_1 \dot{\epsilon}_{ij}^{pl}
\]  

(104.645)

where \(a_1\) is a hardening constant. The hardening constant \(a_1\) is denoted as **kinematic hardening rate** in Table 104.4.

If only linear kinematic hardening (Equation 104.645) is assumed, the back stress \(\alpha_{ij}\) is expressed explicitly, and can be substituted into Equation 104.627 yielding:

\[
\omega_{pl} = \omega_{pl}^{kin} = \frac{a_1}{2\rho} \dot{\epsilon}_{ij}^{pl} \dot{\epsilon}_{ij}^{pl}
\]  

(104.646)

Take the time derivative of the above equation:

\[
\dot{\omega}_{pl} = \frac{a_1}{\rho} \dot{\epsilon}_{ij}^{pl} \dot{\epsilon}_{ij}^{pl}
\]  

(104.647)
Then the rate of dissipation due to plasticity can be rewritten as:

\[ \Phi = \sigma_{ij} \dot{\varepsilon}_{ij}^{pl} - \rho \dot{\psi}_{pl} = (s_{ij} - \alpha_{ij}) \dot{\varepsilon}_{ij}^{pl} = k m_{ij} \dot{\varepsilon}_{ij}^{pl} \]  

(104.648)

Notice that the term \( m_{ij} \dot{\varepsilon}_{ij}^{pl} \) denotes the magnitude of the rate of plastic strain. Since only linear kinematic hardening is assumed, the internal variable \( k \) will remain constant. So if loads are applied in such a way that the rate of plastic strain is constant, then the rate of dissipation will also remain constant. In other words, the accumulated dissipation will be linearly increasing under the assumption of linear kinematic hardening.

Figure 104.84 shows stress–strain response (left) and energy computation results (right) of an elastic-plastic material modeled using von Mises plasticity with linear kinematic hardening.

As expected, the plastic dissipation increases linearly once the material yields. In contrast to the isotropic hardening case, a significant amount of the input work is dissipated due to material plasticity. The ratio of dissipated energy to input work is largely influenced by the material parameters. However, in general, energy dissipation will be observed if kinematic hardening model is used.

Another important observation is that the plastic work decreases during certain phases of reverse loading, while the actual rate of energy dissipation is always nonnegative. It is important to distinguish plastic work from plastic energy dissipation. Otherwise, one might argue that accumulated energy dissipation can increase or decrease, which is a common mistake observed in a number of publications that violates the second law of thermodynamics.
Armstrong-Frederick Kinematic Hardening. Armstrong-Frederick kinematic hardening model Armstrong and Frederick (1966) is often used to simulate elastic-plastic material behavior under cyclic loading. Material parameters of the Armstrong-Frederick kinematic hardening rule can be derived from basic thermodynamics. The following equation is a general expression for Armstrong-Frederick kinematic hardening rule:

\[ \dot{\alpha}_{ij} = a_1 \dot{\varepsilon}^{pl}_{ij} - a_2 \dot{\lambda} \alpha_{ij} \]  

(104.649)

where \( \dot{\lambda} \) is a non-negative scalar plastic multiplier and \( a_2 \) is a non-negative material hardening constant. It can be proven that \( a_1/a_2 \) is related to the limit of back stress magnitude \( |\alpha_{ij}| \). In Table 104.4, the hardening constants \( a_1 \) and \( a_2 \) correspond to parameters armstrong_frederick_ha and armstrong_frederick_cr.

Taking the time derivative of the kinematic part of plastic free energy (Equation 104.647), and substituting the expression of back stress \( \alpha_{ij} \) (Equation 104.649) gives:

\[ \dot{\psi}_{pl}^{kin} = \frac{1}{\rho a_1} \alpha_{ij} \dot{\alpha}_{ij} = \frac{1}{\rho} \alpha_{ij} (\dot{\varepsilon}^{pl}_{ij} - \frac{a_2}{a_1} \dot{\lambda} \alpha_{ij}) \]  

(104.650)

Then the rate of plastic energy dissipation of an Armstrong-Frederick kinematic hardening elastic-plastic material is given by:

\[ \Phi = \sigma_{ij} \dot{\varepsilon}^{pl}_{ij} - \rho \dot{\psi}_{pl}^{kin} = s_{ij} \dot{\varepsilon}^{pl}_{ij} - \alpha_{ij} \dot{\varepsilon}^{pl}_{ij} + \frac{a_2}{a_1} \dot{\lambda} \alpha_{ij} \alpha_{ij} = k m_{ij} \dot{\varepsilon}^{pl}_{ij} + \frac{a_2}{a_1} \dot{\lambda} \alpha_{ij} \alpha_{ij} \]  

(104.651)

Compared with Equation 104.648, the above expression has an additional term which makes the rate of plastic dissipation non-constant even if the rate of plastic strain is constant. As the back stress \( \alpha_{ij} \) becomes larger when load increases, the rate of plastic dissipation also increases. This indicates a nonlinear result of total plastic dissipation, which is exactly what we have observed in our computations.

Figure 104.85 shows the energy computation results of an elastic-plastic material modeled using von Mises plasticity with Armstrong-Frederick kinematic hardening. Compared to all previous cases, the material response of this model is more sophisticated and more realistic. Decrease of plastic work is observed, again, while the plastic dissipation is always nonnegative during the entire simulation. For both linear and nonlinear kinematic hardening cases, the plastic free energy is relatively small compared to the plastic dissipation.

104.11.3.3 Drucker–Prager Plasticity

It has been proven that von Mises plasticity generally performs poorly in modeling pressure-sensitive materials like soils. In this section, the thermomechanical formulations presented in earlier sections are
Figure 104.85: Energy analysis of elastic-plastic material modeled using von Mises plasticity with Armstrong–Frederick kinematic hardening: (a) Stress–strain curve; (b) Input work, plastic dissipation, strain energy, and plastic work.

Table 104.5: Model parameters for cases using Drucker–Prager plasticity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Hardening Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Linear Isotropic</td>
</tr>
<tr>
<td>mass_density</td>
<td>kg/m$^3$</td>
<td>2000</td>
</tr>
<tr>
<td>elastic_modulus</td>
<td>MPa</td>
<td>150</td>
</tr>
<tr>
<td>poisson_ratio</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>druckerprager_k</td>
<td></td>
<td>0.25</td>
</tr>
<tr>
<td>confining_stress</td>
<td>kPa</td>
<td>100</td>
</tr>
<tr>
<td>isotropic_hardening_rate</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>kinematic_hardening_rate</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>armstrong_frederick_ha</td>
<td>MPa</td>
<td></td>
</tr>
<tr>
<td>armstrong_frederick_cr</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
applied to models using Drucker–Prager yield criteria with different hardening types. The material model parameters used in this section are summarized in Table 104.5.

The yield function of Drucker–Prager plasticity is:

\[
    f = [(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})]^{0.5} - \frac{\sqrt{2}}{3} kp
\]

where \( p = (1/3)\delta_{ij}\sigma_{kk} \) is the mean stress (or hydrostatic pressure). Note that in this form of Drucker–Prager plasticity, the internal variables \( \alpha_{ij} \) and \( k \), as well as the hardening constants \( \kappa_1 \) and \( a_1 \), are dimensionless.

For the computation of plastic free energy in Drucker–Prager plasticity models, Equation 104.627 is modified:

\[
\psi_{iso}^{pl} = \frac{1}{2}\rho \left( \kappa_1 p \right)^2 = \frac{1}{2}\rho \kappa_1 k^2 p
\]

\[
\psi_{kin}^{pl} = \frac{1}{2}\rho \left( a_1 p \right) \left( \alpha_{ij} p \right) \left( \alpha_{ij} p \right) = \frac{1}{2}\rho a_1 \alpha_{ij} \alpha_{ij} p
\]

All examples presented in this section are using non-associated Drucker–Prager plasticity. The plastic potential function is of von Mises type so that only deviatoric plastic flow exists. In addition, all cases are loaded with constant hydrostatic pressure, which means that the plastic flow direction \( m_{ij} \) is the same as the direction of the total plastic strain \( \varepsilon_{ij}^{pl} \). With the above conditions, Equation 104.639, 104.644, and 104.648 are all still valid for the following examples, as will be observed in their results.

It should be noted that the proposed energy computation approach can also be applied to associated Drucker–Prager plasticity and non-associated Drucker–Prager plasticity with different plastic potential functions. The loading condition can be arbitrary, even with evolving hydrostatic pressures.

**Linear Isotropic Hardening.** Figure 104.86 shows the stress–strain response and energy computation results of an elastic-plastic material modeled using Drucker–Prager plasticity with linear kinematic hardening.

No plastic dissipation is observed in this case, which have been theoretically proven in Equation 104.644. This example again indicates that isotropic hardening is not capable of proper modeling of energy dissipation in elastic-plastic materials.

**Prager Linear Kinematic Hardening.** Figure 104.87 shows the stress–strain response and energy computation results of an elastic-plastic material modeled using Drucker–Prager plasticity with linear kinematic hardening.

Plastic dissipation increases linearly when the material yields, while the plastic work decreases during certain phases of reverse loading. This observation is consistent with the theoretical conclusion drawn from Equation 104.648.
Figure 104.86: Energy analysis of elastic-plastic material modeled using Drucker–Prager plasticity with linear isotropic hardening: (a) Stress–strain curve; (b) Input work, plastic dissipation strain energy and plastic work.

Figure 104.87: Energy analysis of elastic-plastic material modeled using Drucker–Prager plasticity with linear kinematic hardening: (a) Stress–strain curve; (b) Input work, plastic dissipation strain energy and plastic work.
Note that the results of the above two examples share high similarity with those of the cases modeled with associated von Mises plasticity. This is because the hydrostatic pressures were constant during shearing, which makes the pressure-dependent feature of Drucker–Prager plasticity not observed. The energy computation results are expected to be more complicated with different loading conditions. However, the difference between plastic work and plastic dissipation will be observed. And the incremental plastic dissipation should always be nonnegative.

Armstrong–Frederick Kinematic Hardening. In order to illustrate the influence of hydrostatic pressure to the energy dissipation in Drucker–Prager models, three cases with different confining pressures are studied. Armstrong–Frederick kinematic hardening is used here to model the nonlinear hardening response of pressure-dependent material, like soils. Figure 104.88 shows the stress–strain response and energy computation results of these three cases.

As can be observed from Figure 104.88, the slope of stress–strain loop increases, which means the material becomes stiffer, as the confining stress increases. Also, the size of elastic region becomes larger when the confining stress is bigger.

Plastic dissipation and plastic free energy starts to evolve as soon as the material yields. The pattern of evolution of energy components are the same for all three cases, while the value of plastic dissipation decreases as the confining stress increases. This is expected since the material becomes stiffer and harder to plastify with a higher confining stress.

104.11.4 Conclusions

Presented was a methodology for (correct) computation of energy dissipation in elastic-plastic materials based on the second law of thermodynamics. A very important role of plastic free energy was analyzed, with highlights on its physical nature and theoretical formulations. The proposed methodology has been illustrated using a number of elasto-plastic material models.

An analysis of a common misconception that equates plastic work and dissipation, which leads to the violation of the basic principles of thermodynamics, was addressed. A conceptual example, for granular materials, was used to explain the physical meaning of plastic free energy. It was also shown that plastic free energy is responsible for the evolution of internal variables.

It was shown that energy balance is ensured by taking into consideration all energy components, including kinetic and strain energy. Input work was balanced with the stored and dissipated energy, expressed as the summation of all possible components.

Presented approach was illustrated and tested using several elastic-plastic constitutive models with various hardening rules. Elastic materials showed no energy dissipation (as expected), leading to the
Figure 104.88: Energy analysis of elastic-plastic material modeled using Drucker–Prager plasticity with Armstrong–Frederick kinematic hardening: (a) Confining stress = 100 kPa; (b) Confining stress = 300 kPa; (c) Confining stress = 500 kPa;
input work being equal to the stored energy. Elastic-perfectly plastic materials had no change in plastic free energy, which led to the equality of plastic work and plastic dissipation and indicated no evolution of particle arrangements. The plastic dissipation, in that case, was observed to be increasing linearly. Isotropic hardening materials experienced zero dissipation even after yielding. This observation was surprising, but verified by further derivation of energy equations. This observation also serves as a reminder that the isotropic hardening rules can be used, but only with observed lack of energy dissipation. Prager’s linear and Armstrong-Frederick nonlinear kinematic hardening materials both gave significant dissipations, with large fluctuation of plastic free energy as well. In the case with linear kinematic hardening, linear increase of dissipation was derived and observed, while energy was dissipated nonlinerly in the case of nonlinear kinematic hardening. Although the plastic free energy was not significant for some materials, it is noted that it should always be recognized and considered during energy analysis, so that the basic principles of thermodynamics are maintained.

104.12 Energy Dissipation Calculations for Structures

This section is based on Yang et al. (2019)

104.12.1 Introduction

Mechanical energy in soil structure interaction (SSI) systems are dissipated during the irreversible dissipative process of energy transformation in which entropy of the system increases. Energy dissipation has been used, directly or indirectly, as a key parameter to evaluate damage in elastic-plastic materials. A common misconception of plastic work and energy dissipation due to plasticity has been noticed in a number of publications Uang and Bertero (1990), Léger and Dussault (1992), Symans and Constantinou (1998), Soong and Spencer (2002), Symans et al. (2008), Wong (2008), Nehdi et al. (2010) in which violations of the second law of thermodynamics were observed. As presented in an earlier section (and also by Yang et al. (2018), the correct formulation for energy analysis on elastic-plastic solids has been derived from the second law of thermodynamics. The theoretical and computational framework has been verified through system energy balance in a series of numerical studies on elastic and elastic-plastic material models. The purpose of this section is to present a methodology of correctly evaluating energy dissipation in structural elements, which is crucial in determining the safety and economy of a SSI system.

It has been shown Dafalias et al. (2002), Feigenbaum and Dafalias (2007), Yang et al. (2018) that the difference between plastic work and plastic dissipation is the plastic free energy, or cold work, which can be calculated from material internal variables (or state variables), like radius of yield surface or back stress. This computation can be easily performed on solids modeled with classic elasticity/plasticity
constitutive relationships in which internal variables are computed and obtained at every time step. On the other hand, constitutive relationships used to model nonlinear structural elements were proposed mainly based on experimental results Spacone et al. (1996a), Spacone et al. (1996b), Lee and Fenves (1998), Popovics (1973), Mander et al. (1988), Chang and Mander (1994), Waugh (2009), Kolozvari et al. (2015). The internal variables used in these models are different than those used in classic constitutive models for solids. Therefore, a new methodology that can correctly evaluate energy storage and dissipation in structural elements is required.

During the recent few decades, a number of studies have been conducted with focus on energy analysis of SSI systems Uang and Bertero (1990), Léger and Dussault (1992), Kalkan and Kunnath (2007), Kalkan and Kunnath (2008), Symans et al. (2008), Gajan and Saravanathiiban (2011), Moustafa (2011), Moustafa and Mahmoud (2014), Mezgebo and Lui (2017), Deniz et al. (2017). Despite different formulations used, the calculations of energy dissipation due to hysteretic damping (material elasto-plasticity) in these publications were all performed without consideration of plastic free energy, which lead to violations of principles of thermodynamics. It is worth pointing out that such oversight is not rare, especially in literature of civil and geotechnical engineering.

Early work reported by Farren and Taylor (1925) and Taylor and Quinney (1934) showed that plastic free energy could be significant in metals, thus should not be neglected without reasoning. The ratio of plastic work converted into heat, usually referred to as the Quinney–Taylor coefficient, was measured to be between 0.6 to 1.0 Belytschko et al. (1991), Zhou et al. (1996), Dolinski et al. (2010), Osovski et al. (2013). Mason Mason et al. (1994) pointed out that the the Quinney–Taylor coefficient is both strain and strain rate dependent but could be assumed to be a constant in most cases. A constitutive model for metals was presented by Rosakis et al. Rosakis et al. (2000), Hodowany et al. (2000), Ravichandran et al. (2002) based on thermoplasticity, which can model the evolution of energy dissipation and has been validated through experiments. Semnani et al. (2016) presented a thermoplastic framework that could predict strain localization in transversely isotropic materials.

Despite of the existence of sophisticated theories that are capable of modeling the evolution of energy dissipation, including those mentioned earlier, most constitutive relationships used to model structural elements do not involve thermodynamics or thermoplasticity. One commonly used finite element (FE) technique of modeling frame structures is fiber section, in which beams and columns are divided into multiple uniaxial fibers with various constitutive models. This model have been proved to be able to capture nonlinear stress–strain behaviors of structural elements under axial loading and/or bending.

Problems arise when such elements are used to calculate energy dissipation. As observed in many publications Kwan and Billington (2001), Zhu et al. (2006), Gajan and Saravanathiiban (2011), Wang et al. (2012), Zhang et al. (2013), Nikbakht et al. (2014) energy analysis were performed based on the
hysteretic stress–strain or force–displacement response of the elements. This indicates that plastic work was confused with plastic energy dissipation, which is the common misconception pointed out earlier. It is also important to point out that various damage indices derived from energy dissipation are used widely to evaluate damage in structures. Such parameters will not be valid if the fundamental formulation of energy dissipation is incorrect.

In order to correctly evaluate energy dissipation in structural elements modeled with fiber sections, the framework of thermo-mechanics must be enforced on the uniaxial constitutive models. Focus of this section is on proper modeling of different forms of energy (storage and dissipation) in uniaxial materials that follows the second law of thermodynamics. Theoretical and computational formulations of energy dissipation in uniaxial concrete and steel fibers are presented. A series of FE simulations are carried out using the Real-ESSI (Jeremić et al., 1988-2021) to illustrate the energy behavior of structural systems. The method is verified by comparing the input work and the energy storage and dissipation in the system. The difference between accumulated plastic work and accumulated plastic dissipation, which can be significant in many cases, is addressed. Finally, conclusions on plastic energy dissipation in structural elements are drawn from the verified results.

104.12.2 Theoretical and Computational Formulations

104.12.2.1 Thermomechanical Framework

The theories of continuum thermo-mechanics have been discussed in a number of earlier publications Lubliner (1972), Rosakis et al. (2000), from which the fundamental framework of this study is derived. General equations of elastoplasticity and thermodynamics are modified with a few plausible assumptions to accommodate the scope of this study. Small deformation theory is assumed, so that the small strain tensor \( \epsilon_{ij} \) is used to describe deformation of the material body. All equations in this section are expressed in index notation.

The general thermomechanical process is governed by momentum balance and the first and second law of thermodynamics. The localized version of the first law of thermodynamics (energy balance equation) is given in the form:

\[
\sigma_{ij} \dot{\epsilon}_{ij} + q_{i,i} + \rho r = \rho \dot{e}
\]  

(104.655)

where the term \( \sigma_{ij} \epsilon_{ij} \) is called the stress power, \( q_{i} \) are the components of the heat flux vector, \( \rho \) is the mass density of the material, \( r \) is the heat supply per unit volume, and \( e \) is the internal energy per unit volume. Note that in this section all stresses are defined as effective stresses. In order to avoid confusion, the common notation \( (\sigma'_{ij}) \) will not be used. Standard definition of stress from mechanics of materials, i.e. positive in tension, is used.
The localized version of the second law of thermodynamics (Clausius–Duhem inequality) is expressed as:

\[ \rho \dot{\eta} - \left( \frac{q_i}{\theta} \right)_i - \frac{1}{\theta} \rho r \geq 0 \]  \hspace{1cm} (104.656)

where \( \eta \) is the entropy per unit volume and \( \theta \) is the absolute temperature.

Substituting the heat supply per unit volume \( r \) in Equation 104.656 with the expression from Equation 104.655, and introducing the rate of change of internal dissipation per unit volume \( \Phi \) gives:

\[ \rho \theta \dot{\eta} - \rho \dot{\psi} + \sigma_{ij} \dot{\epsilon}_{ij} + \frac{1}{\theta} q_i \theta \dot{i}_i = \Phi + \frac{1}{\theta} q_i \theta \dot{i}_i \geq 0 \]  \hspace{1cm} (104.657)

Note that the internal dissipation can have many sources, including material plasticity, viscous coupling, radiation damping, and other forms of energy dissipation.

The Helmholtz free energy per unit volume \( \psi \), which is referred to as free energy in this section, is defined as:

\[ \psi = e - \theta \eta \]  \hspace{1cm} (104.658)

The second law of thermodynamics can be expressed in terms of free energy \( \psi \) as:

\[ \Phi + \frac{1}{\theta} q_i \theta \dot{i}_i = -\rho \dot{\psi} - \rho \dot{\theta} \eta + \sigma_{ij} \dot{\epsilon}_{ij} + \frac{1}{\theta} q_i \theta \dot{i}_i \geq 0 \]  \hspace{1cm} (104.659)

The rate of internal dissipation per unit volume \( \Phi \) can be written as:

\[ \Phi = \sigma_{ij} \dot{\epsilon}_{ij} - \rho \dot{\psi} - \rho \dot{\theta} \eta \]  \hspace{1cm} (104.660)

At this point, a few assumptions are introduced to simplify the governing equations. According to Feigenbaum and Dafalias (2007), Collins and Houlsby (1997), Collins (2002), Collins and Kelly (2002), it can be assumed that the deformation of soil and structural elements under earthquake loading is approximately isothermal, which indicates that the temperature field \( \theta \) is constant and uniform. This approximation is reasonable considering the fact that seismic energy is mostly carried by the low-frequency components of earthquake ground motion, which allows the heat generated in the materials to be largely dissipated. With this assumption, the rate of internal dissipation \( \Phi \) is simplified into the form:

\[ \Phi = \sigma_{ij} \dot{\epsilon}_{ij} - \rho \dot{\psi} \geq 0 \]  \hspace{1cm} (104.661)

Next, all material models studied in this section is assumed to be decoupled, which means that the (small) strain tensor can be additively decomposed into elastic and plastic parts:

\[ \epsilon_{ij} = \epsilon_{ij}^{el} + \epsilon_{ij}^{pl} \]  \hspace{1cm} (104.662)
Lubliner (1972) and Collins and Houlsby (1997) showed that this assumption can be deduced if the instantaneous elastic moduli of a material are independent of the internal variables. Under the assumption of decoupled material, the free energy $\psi$ can also be decomposed into elastic and plastic parts as follows:

$$\psi = \psi_{el} + \psi_{pl}$$  \hspace{1cm} (104.663)

where the elastic part of the free energy $\psi_{el}$ is also known as the elastic strain energy, which is defined in incremental form as:

$$\dot{\psi}_{el} = \sigma_{ij} \dot{\epsilon}_{ij}$$  \hspace{1cm} (104.664)

By substituting Equation 104.662, Equation 104.663, and Equation 104.664 into Equation 104.661, the rate of internal dissipation $\Phi$ can be expressed in terms of the rate of plastic free energy $\dot{\psi}_{pl}$:

$$\Phi = \sigma_{ij} \dot{\epsilon}_{ij} - \sigma_{ij} \dot{\epsilon}_{el}^{ij} - \rho \dot{\psi}_{pl} \geq 0$$  \hspace{1cm} (104.665)

Equation 504.8 represents two basic principles that should always be upheld in any energy analysis for decoupled material undergoing isothermal process:

- The stress power that is input into a material body by external loading is transformed into elastic strain energy, plastic free energy, and material internal dissipation. All forms of energy must be considered to maintain energy balance of the material body. This principle ensures the first law of thermodynamics.

- The rate of change of material internal dissipation (plastic dissipation) is nonnegative at any time. In other words, accumulated internal dissipation can not decrease during any time period. This principle ensures the second law of thermodynamics.

Note that material internal dissipation can have many sources. Our interest is the energy dissipation caused by material plasticity, so the term plastic dissipation will be used instead, which indicates no other source of energy dissipation is present in the examples that are being analyzed in the remaining part of this section.

### 104.12.2.2 Plastic Free Energy

The physical nature of plastic free energy is associated with the material micro-structure. For particulate material, like soil, plastic free energy will be accumulated or released if there is evolution of particle arrangement (micro-fabric), which generally happens as soon as the material body is loaded. For other
structural and geotechnical materials, like metals, their micro-structures are represented by the shape and arrangement of the crystals, whose evolution will result in change in plastic free energy. Detailed explanations of the evolution of plastic free energy can be found in publications by Besseling and Van Der Giessen (1994), Collins and Kelly (2002), and Yang et al. (2018).

Using Equation 504.8, the energy dissipation of any elastic-plastic material under isothermal loading process can be calculated, if all the terms on the right hand side of the equation is known. For most elastic-plastic constitutive models, the stress tensor $\sigma_{ij}$ and the elastic strain tensor $\epsilon_{ij}^{el}$ are being calculated as simulation progresses. The challenging task is to evaluate the plastic free energy term $\psi_{pl}$, whose formulation depends on the internal variables used in the constitutive model.

For a decoupled elastic-plastic material model that exhibits both isotropic and kinematic hardening, the plastic free energy is decomposed into isotropic and kinematic parts, which are calculated individually and then added up. The formulation of plastic free energy in this type of material was given by Feigenbaum and Dafalias (2007):

$$\psi_{pl} = \psi_{iso}^{pl} + \psi_{kin}^{pl} = \frac{1}{2 \rho k_1} k^2 + \frac{1}{2 \rho a_1} \alpha_{ij} \alpha_{ij}$$  \hspace{1cm} (104.666)

where $\psi_{iso}^{pl}$ and $\psi_{kin}^{pl}$ are the isotropic and kinematic parts of the plastic free energy, respectively, $k$ is the radius of yield surface, $\alpha$ is the back stress, $k_1$ and $a_1$ are non-negative material constants. Note that Equation 104.666 can be used for a wide range of constitutive models with various yield functions, including von Mises and Drucker-Prager yield criteria whose energy behaviors has been studied and presented by Yang et al. (2018). Such materials are usually used to model soils and parts of the structure that need to be modeled with solid elements in SSI system.

However, frame structures are generally modeled with beam-column elements in combination with fiber sections and uniaxial material models, where Equation 104.666 does not apply. It should be realized that most uniaxial constitutive relationships are capable of modeling the stress–strain behavior but not the energy dissipation features of the material. Therefore, an approach that follows our thermomechanical framework is presented to correctly evaluate energy storage and dissipation in these material models.

### 104.12.2.3 Energy Dissipation in Beam-Column Element

Beams and columns are modeled with nonlinear displacement-based beam element, which is implemented in the Real-ESSI Simulator. In order to incorporate confined/unconfined concrete and steel reinforcement bars into beam elements, fiber sections are constructed with corresponding uniaxial fibers. A bottom-fixed reinforced concrete column model is shown in Figure 104.89, along with the constitutive relationships used for the concrete and steel fibers.
Uniaxial Steel Fiber. The uniaxial steel material model examined in this study was developed by Menegotto and Pinto (1973) and extended by Filippou et al. (1983), and is capable of capturing the nonlinear hysteretic behavior and isotropic strain-hardening effect of steel. The stress–strain response of this material is shown in Figure 104.90, along with some of the material parameters. The model, as presented in Menegotto and Pinto (1973), takes on the form:

\[
\sigma^* = b \epsilon^* + \frac{(1 - b) \epsilon^*}{(1 + \epsilon^* R)^1/R}
\]

with

\[
\epsilon^* = \frac{\epsilon - \epsilon_r}{\epsilon_0 - \epsilon_r}; \quad \sigma^* = \frac{\sigma - \sigma_r}{\sigma_0 - \sigma_r}
\]

where \(b\) is the strain-hardening ratio, \(\epsilon_r\) and \(\sigma_r\) are the strain and stress at the point of strain reversal, \(\epsilon_0\) and \(\sigma_0\) are the strain and stress at the point of intersection of the two asymptotes, \(R\) is the curvature parameter that governs the shape of the transition curve between the two asymptotes. Note that this model is for uniaxial materials, in which the stresses and strains are scalars instead of tensors.

The expression for the curvature parameter \(R\) is suggested by Menegotto and Pinto (1973):

\[
R = R_0 - \frac{c_{R_2} \xi}{c_{R_2} + \xi}
\]

where \(R_0\) is the value of the curvature parameter \(R\) during initial loading, \(c_{R_1}\) and \(c_{R_2}\) are degradation parameters that need to be experimentally determined. The parameter \(\xi\), that is updated after strain

Figure 104.89: Schematic of a bottom-fixed column modeled with concrete and steel fibers.
Figure 104.90: Constitutive model for uniaxial steel fiber (Menegotto and Pinto (1973)).

reversal, is defined as:

\[
\xi = \frac{|(\epsilon_m - \epsilon_0)|}{\epsilon_y}
\]

where \(\epsilon_m\) is the maximum (or minimum) strain at the previous strain reversal point, depending on the loading direction of the material. If the current incremental strain is positive, the parameter \(\epsilon_m\) takes the value of the maximum reversal strain. Parameter \(\epsilon_y\) is the monotonic yield strain.

In order to capture isotropic hardening behavior, Filippou et al. (1983) introduced stress shift mechanism into the original model by Menegotto and Pinto (1973). Note that the hardening rate in compression and tension can be different by choosing different hardening parameters for compression and tension. The proposed relation takes the form:

\[
\frac{\sigma_{st}}{\sigma_y} = a_1 \left( \frac{\epsilon_{\text{max}}}{\epsilon_y} - a_2 \right)
\]

where \(\sigma_{st}\) is the shift stress that determines the shift of yield asymptote, \(\epsilon_{\text{max}}\) is the absolute maximum strain at strain reversal, and \(a_1\) and \(a_2\) are hardening parameters in compression that are experimentally determined. In the case of tension, the hardening parameters \(a_1\) and \(a_2\) in Equation 104.671 are changed to \(a_3\) and \(a_4\) that are also determined by experiment or obtained from the literature.
The energy computation procedure for this uniaxial steel model is shown in Figure 507.4, and it follows the thermomechanical framework established earlier in this section.

![Energy computation of uniaxial steel fiber](image)

Figure 104.91: Energy computation of uniaxial steel fiber: (a) Monotonic loading branch; (b) Cyclic loading branch.

Note that the only difference between the monotonic loading branch (Figure 507.4(a)) and the cyclic loading branch (Figure 507.4(b)) is that the strain reversal point $c$ is at the origin $o$ in the monotonic case. So the following explanation of the proposed energy computation method applies to both monotonic and cyclic loading scenarios.

Firstly, the elastic strain energy density $E_S$ is defined in accordance with the classic assumption that it is only a function of current stress state of the material, which yields:

$$E_S = E_S(\sigma) = \frac{1}{2E_0}\sigma^2$$

(104.672)

Graphically, the elastic strain energy density of the material shown in Figure 507.4 at states $a$ and $b$ are the triangular areas $afd$ and $bge$. Then the incremental form of Equation 507.9 is simply:

$$dE_S = \frac{1}{E_0}\sigma d\sigma$$

(104.673)

Next, the incremental plastic dissipation density $D_P$ from state $a$ to $b$ is assumed to be the triangular area $abc$:

$$dD_P = \frac{1}{2}[(\sigma - \sigma_r)d\epsilon - (\epsilon - \epsilon_r)d\sigma]$$

(104.674)

This assumption ensures that the incremental plastic dissipation is nonnegative, which is one of the two basic principles of our thermomechanical framework. One special case is when the material exhibits
no cyclic softening, which means that perfectly overlapping stress–strain loops will be observed, the
energy dissipation calculated using Equation 507.11 for one cyclic is the area of the hysteresis loop. In
the proposed thermomechanical framework, the area of hysteresis loop should be equal to the plastic
work, rather than plastic dissipation, done in one loading cycle. But in this special case of no cyclic
softening, which means no evolution of material state and thus no development of plastic free energy
after a complete loading cycle, the plastic work equals to the plastic dissipation in the material in one
loading cycle.

For general case where the material does exhibit cyclic softening, plastic free energy density $E_P$
is graphically denoted by the areas $adoc$ and $beoc$ at states $a$ and $b$, respectively. The formulation
representing this assumption is given by:

$$E_P = \frac{1}{2} \left[ \sigma \left( \epsilon - \frac{\sigma}{E_0} - \epsilon_r \right) + \sigma_r \epsilon \right]$$

(104.675)

The incremental form of Equation 507.12 is:

$$dE_P = \frac{1}{2} \left[ (\sigma + \sigma_r) d\epsilon + \left( \epsilon - \frac{1}{E_0} \sigma - \epsilon_r \right) d\sigma \right]$$

(104.676)

Adding Equation 507.10, 507.11, and 507.13, the incremental form of energy balance is achieved:

$$dE_S + dE_P + dD_P = \sigma d\epsilon$$

(104.677)

where the increment of three energy components add up to the increment of stress power during any
loading step.

Uniaxial Concrete Fiber. The uniaxial concrete material model used in this study is based on the
model proposed by Yassin (1994), which is capable of modeling the nonlinear hysteretic behavior and
damage effect of concrete. The material parameters and stress–strain response of this material are shown
in Figure 104.92.

The monotonic envelope curve of this model in compression is based on the model of Kent and Park
(1971) and later generalized by Scott et al. (1982). For a given strain $\epsilon_c$, the compressive stress $\sigma_c$ and
corresponding tangent stiffness $E$ are given by:

$$\epsilon_c \leq \epsilon_{cs} \quad \sigma_c = f_{cs} \left[ 2 \left( \frac{\epsilon_c}{\epsilon_{cs}} \right) - \left( \frac{\epsilon_c}{\epsilon_{cs}} \right)^2 \right] \quad E = E_c \left( 1 - \frac{\epsilon_c}{\epsilon_{cs}} \right)$$

(104.678)

$$\epsilon_{cs} < \epsilon_c \leq \epsilon_{cu} \quad \sigma_c = \frac{\epsilon_c - \epsilon_{cs}}{\epsilon_{cu} - \epsilon_{cs}} (f_{cu} - f_{cs}) + f_{cs} \quad E = f_{cu} - f_{cs}$$

(104.679)

$$\epsilon_c > \epsilon_{cu} \quad \sigma_c = f_{cu} \quad E = 0$$

(104.680)

where $f_{cs}$ is the maximum compressive strength of the concrete material, $\epsilon_{cs}$ is the concrete strain at
compressive strength, $f_{cu}$ is the ultimate (crushing) strength of the concrete material, $\epsilon_{cu}$ is the concrete
strain at ultimate strength, and $E_c$ is the initial concrete tangent stiffness that can be calculated using the equation:

$$E_c = \frac{2f_{cs}}{\varepsilon_{cs}}$$  \hspace{1cm} (104.681)

All material parameters should be determined by experiment or related literature data.

The cyclic behavior of this concrete model in compression is shown in Figure 104.92. One assumption of this model is that all reloading lines intersect at a common point, where the stress $\sigma_r$ and strain $\varepsilon_r$ are given by the following expressions:

$$\varepsilon_r = \frac{f_{cu} - \lambda E_c \varepsilon_{cu}}{E_c(1 - \lambda)}$$  \hspace{1cm} (104.682)

$$\sigma_r = E_c \varepsilon_r$$  \hspace{1cm} (104.683)

After unloading from a point on the compressive monotonic envelope, the model response is bounded by two lines that are defined by:

$$\sigma_{max} = \sigma_m + E_r(\varepsilon_c - \varepsilon_m)$$  \hspace{1cm} (104.684)

$$\sigma_{min} = 0.5E_r(\varepsilon_c - \varepsilon_t)$$  \hspace{1cm} (104.685)
where
\[ E_r = \frac{\sigma_m - \sigma_r}{\epsilon_m - \epsilon_r} \]  (104.686)
\[ \epsilon_t = \epsilon_m - \frac{\sigma_m}{E_r} \]  (104.687)

where \( \sigma_m \) and \( \epsilon_m \) are the stress and strain at the unloading point on the compressive monotonic envelope, respectively. If the unloading–reloading cycle is incomplete, the material response will be a straight line with slope \( E_c \), as shown in Figure 104.92.

The tensile behavior of this concrete model considers tension stiffening and the effects of initial cracking. Details of the monotonic and cyclic behavior of this model under tensile stress can be found in Yassin (1994).

Since there are different loading/unloading branches in this model, the energy computation needs to be considered separately for each branch. One energy component that remains the same in all loading cases is the elastic strain energy density \( E_S \), which is only a function of current stress:
\[ E_S = E_S(\sigma) = \frac{1}{2} E_c \sigma^2 \]  (104.688)
And the incremental form of Equation 104.688 is:
\[ dE_S = \frac{1}{E_c} \sigma d\sigma \]  (104.689)

In order to calculate plastic dissipation, a few assumptions are made that ensures the energy behavior of the concrete material to follow the proposed thermomechanical framework:

- Majority of energy is dissipated during first monotonic load.
- Subsequent cycles of loading, on an already damaged concrete, do not dissipate much energy.
- No energy is dissipated during unloading in both compressive and tensile conditions.
- When the material is cyclically loaded under compression, energy dissipation only happens when the stress reaches the upper bound \( \sigma_{max} \).
- No energy is dissipated during cyclic loading when the material is under tension.

For a single loading step from stress state \( a \) to \( b \) in each subplot of Figure 104.93, the energy dissipation is represented by the shaded area.

If the material is under compression (Figure 104.93 (a), (b), and (c)), the amount of energy dissipated in the concrete fiber \( D_P \) is calculated by taking the area \( abcdef \), which is generally a hexagon formed by the two unloading paths originated from stress state \( a \) and \( b \):
\[ dD_P = \frac{1}{2} [ (\sigma - \sigma_c) d\epsilon + (\epsilon_c - \epsilon) d\sigma + (\epsilon_c - \epsilon_f) \sigma + (\sigma_f - \sigma_c)(\epsilon - \epsilon_t) + \sigma_c d\epsilon_t ] \]  (104.690)
where the stress and strain at point \( f \) can be computed using the following expression:

\[
\epsilon_f = \frac{\sigma + 0.5E_r\epsilon_t - E_c\epsilon}{0.5E_r - E_c} \quad \text{and} \quad \sigma_f = 0.5E_r(\epsilon_f - \epsilon_t) \tag{104.691}
\]

Point \( c \) can be calculated using the same fashion, but with all variables evaluated at state \( b \).

Note that the hexagon becomes quadrilateral in the cases of cyclic loading within the monotonic envelope, as can be observed in Figure 104.93 (b) and (c). But Equation 104.690 and 104.691 remains valid, obviously.

Plastic free energy \( E_P \) in this concrete material is calculated by taking the triangular area \( fge \) at state \( a \):

\[
E_P = \frac{1}{2} \left[ \left( \epsilon - \frac{\sigma}{E_c} - \epsilon_t \right) \sigma_f \right] \tag{104.692}
\]
The incremental form of Equation 104.692 is obtained by taking the difference between the plastic free energy at state \( a \) and \( b \):

\[
dE_P = \frac{1}{2} \left[ \left( \sigma_c - \sigma_f - \frac{1}{E_c} \sigma \right) (\epsilon - \epsilon_t) - (d\epsilon - d\epsilon_t) \sigma_c - \frac{1}{E_c} \sigma c \right]
\]

(104.693)

Adding Equation 104.689, 104.690, and 104.693, the incremental form of energy balance is achieved:

\[
dE_S + dE_P + dD_P = \sigma d\epsilon
\]

(104.694)

where the increment of three energy components add up to the increment of stress power during any loading step.

104.12.3 Numerical Studies

Numerical examples presented in this section are performed using the Real-ESSI Simulator Jeremić et al. (1988-2021). Energy dissipation in numerical models consist of fiber section elements and uniaxial steel/concrete fibers are computed and analyzed.

First, numerical simulation of steel and plain concrete columns under various loading conditions are performed to study the energy behavior of uniaxial steel and concrete material models. Then, a model of reinforced concrete column, which consists of both concrete and steel fibers, is constructed and simulated to illustrate the energy dissipation in realistic structural elements. Finally, a bare steel frame structure is modeled with fiber section elements and loaded with seismic motion. Through these examples, it will be shown that the difference between plastic work and plastic energy dissipation can be significant.

External loads are applied incrementally using displacement-control scheme. System equations are solved using Newton-Raphson iteration algorithm and UMFPACK solver, which are available in Real-ESSI. Static integration algorithm is used for the column cases, while Newmark integration is used for the dynamic steel frame case. Note that viscous and numerical damping are excluded from all cases, in order to accurately evaluate energy dissipation due to material elastoplasticity.

104.12.3.1 Steel Column

In order to verify the proposed energy computation approach for uniaxial steel material model, examples of steel columns are studied in this section. As shown in Figure 104.94, the one-meter-long column model is fixed at the bottom, and loads are applied at the top. The size of the cross section is 100 mm $\times$ 100 mm. The parameters for uniaxial steel material used in this section are summarized in Table 104.6.
Figure 104.94: Schematic of the steel/plain-concrete column modeled with fiber sections and uniaxial steel/concrete materials.

Table 104.6: Material model parameters used in steel column examples.

<table>
<thead>
<tr>
<th>$\sigma_y$ (MPa)</th>
<th>$E$ (GPa)</th>
<th>$b$</th>
<th>$R_0$</th>
<th>$c_{R_1}$</th>
<th>$c_{R_2}$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>413.8</td>
<td>200.0</td>
<td>0.01</td>
<td>18.0</td>
<td>0.925</td>
<td>0.15</td>
<td>0.0</td>
<td>55.0</td>
<td>0.0</td>
<td>55.0</td>
</tr>
</tbody>
</table>

**Cyclic Axial Loading.** Since the fiber material model is uniaxial in nature, axial loading case is being investigated first. The evolution of energy parameters for uniaxial steel material are computed using Equation 507.10, 507.11, and 507.13. Figure 104.95 shows the stress–strain response as well as the energy results of the steel column under cyclic axial loading.

As expected, the stress–strain response shown in Figure 104.95 follows the constitutive model presented in Figure 104.90. Due to the choice of hardening parameters ($a_1$, $a_2$, $a_3$, and $a_4$), isotropic hardening after first loading reversal is relatively small in this case. The evolution of plastic free energy, which is related to the hardening behavior of the constitutive model, is also observed to be insignificant after the first loading reversal. Energy balance in the steel material (Equation 507.14) is maintained during entire simulation.

In this particular case, the difference between plastic dissipation and plastic work is significant during initial loading (or monotonic loading), but then becomes less obvious during cyclic loading, which is
Probably the reason of ignorance of plastic free energy in many studies. It is important to point out that such difference could be significant if different hardening parameters are chosen or complex loading conditions (like seismic loading) are applied.

Another observation made in this example is that the ratio between plastic dissipation and plastic work (the Quinney–Taylor coefficient) changes from 0.5 to 0.9 in just a few loading cycles. Therefore, it is not accurate to prescribe a fixed number to be the Quinney–Taylor coefficient of a material during entire simulation, which is a common assumption made in a number of studies.

**Cyclic Bending Loading.** It has been proven that fiber section elements perform well under axial- and bending-dominant loading conditions. In this case, a cyclic bending moment is loaded on the top of the steel column. Figure 104.96 shows the moment–rotation response as well as the energy results of the steel column under cyclic bending loading.

Clearly, the moment–rotation response and energy results in this case are very similar to those in the axial loading case. When a beam element is under bending, half of the fibers will be under tension while the other half under compression, and the normal stress distribution on any cross section should be symmetric. Since the fiber material model used in this case has almost the same stress–strain response under tension and compression, the energy results in this bending case are expected to share the same pattern with those in the axial loading case.

Note that in both axial and bending cases, the strain energy accumulated in the material body is much smaller than the plastic dissipation. This means that most of the input work results in plastic.
104.12.3.2 Plain Concrete Column

In order to verify the proposed energy computation approach for uniaxial concrete material model, examples of plain concrete columns are studied in this section. The size and setup of the model are the same as those of the steel column, which has been shown in Figure 104.94. The parameters for uniaxial concrete material used in this section are summarized in Table 104.7.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{cs}$</td>
<td>30.2 MPa</td>
</tr>
<tr>
<td>$\epsilon_{cs}$</td>
<td>-0.00219</td>
</tr>
<tr>
<td>$f_{cu}$</td>
<td>6.0 MPa</td>
</tr>
<tr>
<td>$\epsilon_{cu}$</td>
<td>-0.00696</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.5</td>
</tr>
<tr>
<td>$f_{ts}$</td>
<td>3.02 MPa</td>
</tr>
<tr>
<td>$E_t$</td>
<td>5.0 GPa</td>
</tr>
</tbody>
</table>

Monotonic Axial Loading. As stated in the assumptions for energy dissipation in the uniaxial concrete model, the amount of energy dissipated during monotonic loading is much larger than that during unloading/reloading. Such assumption is made based on the brittle nature of concrete materials, in which fracture is the main source of energy dissipation. In this case, the stress–strain response as well as the energy results of the plain concrete column model under monotonic axial compression is investigated and presented in Figure 104.97.

The stress–strain response shown in Figure 104.97 follows the constitutive model presented in Fig-
Figure 104.97: Energy analysis of plain concrete column under monotonic axial loading: (a) Stress–strain response; (b) Plastic dissipation, plastic work, plastic free energy, strain energy, and input work.

As observed in Figure 104.97, large amount of the input work is dissipated during monotonic compression. It is important to point out that the difference between plastic dissipation and plastic work is significant. Plastic free energy starts to accumulate after maximum compressive strength is reached and continue to increase even after crushing. Such behavior can be explained by considering that the micro-structure of concrete continues to evolve as external loads continues to be applied on the material.

The strain energy starts to drop after maximum compressive strength and gradually decreases to almost zero after crushing. This observation is consistent with the fact that the micro-fractures expand rapidly after maximum strength is reached, which leads to the release of elastic strain energy and energy dissipation caused by fracture and crushing.

Cyclic Axial Loading. Due to the complex unloading–reloading rules of the model, the cyclic behavior of the uniaxial concrete material is much more complicated than that of the steel model. Figure 104.98 presents the stress–strain response as well as the energy results of the plain concrete column under cyclic axial loading.

As shown in Figure 104.98, the majority of plastic dissipation happens during monotonic loading branch. Notice that there are drops in plastic work during unloading, but plastic dissipation never decreases, which means that the second law of thermodynamics (Equation 504.8) is always obeyed.

It should be mentioned that there are certain amount of energy dissipation when the material is in...
104.12.3.3 Reinforced Concrete Column

To study the combined influence of concrete and steel fibers, a reinforced concrete column is modeled and tested in this section. The schematic of the model is shown in Figure 104.98, and the material model parameters are summarized in Table 104.8. The cross section of the column is modeled with unconfined concrete, confined concrete, and steel fibers with uniaxial material models discussed in earlier sections.

Table 104.8: Material model parameters used in reinforced concrete column examples.

<table>
<thead>
<tr>
<th>Steel Fiber</th>
<th>Concrete Fiber</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Confined</td>
</tr>
<tr>
<td>$\sigma_y$ (MPa)</td>
<td>413.8</td>
</tr>
<tr>
<td>$E$ (GPa)</td>
<td>200.0</td>
</tr>
<tr>
<td>$b$</td>
<td>0.01</td>
</tr>
<tr>
<td>$R_0$</td>
<td>18.0</td>
</tr>
<tr>
<td>$c_{R_1}$</td>
<td>0.925</td>
</tr>
<tr>
<td>$c_{R_2}$</td>
<td>0.15</td>
</tr>
<tr>
<td>$a_1$, $a_3$</td>
<td>0.0</td>
</tr>
<tr>
<td>$a_2$, $a_4$</td>
<td>55.0</td>
</tr>
</tbody>
</table>
Cyclic Axial Loading. Figure 104.100 shows the force–displacement response as well as the energy results of the reinforced concrete column under cyclic axial loading.

Since concrete fibers have much higher compressive strength than tensile strength, the stress–strain
response of the column is controlled by the concrete part when it is under compression, and by the steel part when under tension. In this case, the initial loading curve clearly resembles the stress–strain response of concrete fiber under monotonic compression. Then the unloading–reloading cycles have the same pattern as those of the steel fiber under cyclic axial loading.

By comparing the energy results shown in Figure 104.100 and those shown in Figure 104.95, it can be seen that the energy dissipation patterns in both cases are very close. This indicates that the majority of input work is dissipated in the steel fibers once the maximum strength of the concrete is exceeded. Again, it can be observed that the difference between plastic work and plastic dissipation is significant in this case.

**Cyclic Bending Loading.** Figure 104.101 shows the moment–rotation response as well as the energy results of the reinforced concrete column under cyclic bending loading.

![Figure 104.101: Energy analysis of reinforced concrete column under cyclic bending loading: (a) Moment–rotation response; (b) Plastic dissipation, plastic work, plastic free energy, strain energy, and input work.](image)

During initial loading, the concrete fibers on the compressive side of the cross section take most of the compression, then during the first reverse loading, the concrete fibers on the other side of the cross section are compressed and damaged. This process is indicated in the moment–rotation curve where a bump caused by the compressive strength of the concrete fibers during the first reverse loading is observed. The energy computation result also shows that the concrete fibers dissipate large amount of energy and get damaged during the first loading cycle. After that, the response of the reinforced concrete column is controlled by the steel bars.
According to the two cases of reinforced concrete column under cyclic loading, the concrete part of the column can dissipate the majority of the input work if the loading is mainly monotonic compression. For cyclic loading cases, if the loading does not exceed the maximum compressive strength of the concrete, which should not be significantly damaged, energy dissipation would be observed in both the concrete and steel. However, if the cyclic loading does exceed the maximum strength of the concrete, the majority of energy dissipation would be in the steel reinforcing bars after the concrete is damaged. This conclusion is consistent with the engineering experience that reinforcements are crucial to the performance of concrete structure during seismic events, when the beams and columns suffer from cyclic loadings.

104.12.3.4 Steel Frame

All the previous cases are assumed to be static or quasi-static to investigate the energy dissipation on material level without the influence of dynamics. In other words, inertia and kinetic energy were not considered. In this example, a steel frame structure is model using fiber section element with uniaxial steel material, as shown in Figure 104.102, and loaded dynamically with a realistic seismic motion. The peak acceleration of the input motion is 0.76 g.

The energy computation results are shown in Figure 104.103. Strain energy, plastic free energy, and plastic dissipation in different stories are computed using Equation [507.10, 507.11, and 507.13]. Input work is computed from the input motion and reaction forces at the base of model. Kinetic energy is computed indirectly by subtracting all other forms of energy from input work.

Note that the kinetic energy of the system is almost zero at certain times, and strain energy reaches a peak value. As can be observed from Figure 104.103, when kinetic energy becomes zero, the combination of strain energy, plastic free energy, and plastic dissipation of the system equals to the total input work. This observation proves that the energy balance of the system is maintained during entire simulation.

At the end of simulation, more than 80% of the total input work is dissipated due to material elasto-plasticity, while about 13% is transformed into plastic free energy which does not result in heating or material damage. In some cases, it might be reasonable to use input work (or energy demand in some literature) as a parameter to evaluate structure safety. However, as shown in this example, correctly computed energy dissipation is more appropriate for evaluation of material damage and structure performance in general.

104.12.4 Conclusions

Presented in this section was a thermodynamic-based methodology for (correct) computation of energy dissipation in nonlinear structural elements modeled with fiber section and uniaxial material models.
Two popular material models for steel and concrete were examined with focus on their nonlinear cyclic behaviors. Formulations for the energy storage and dissipation in these two material models were derived from the basic principles of thermodynamics, in combination with a few reasonable assumptions. The proposed methodology has been illustrated using a series of numerical simulations on columns and frame modeled with fiber section elements.

The misconception between plastic work and plastic dissipation, which leads to the violation of principles of thermodynamics, was addressed. Theoretical derivation and experimental observation have both proven that plastic free energy is a basic form of energy that should not be neglected without proper reasoning. By taking into account of all possible energy forms, including kinetic energy, strain energy, plastic free energy, and plastic dissipation, the first law of thermodynamics (energy balance) was ensured in the proposed methodology.

Physically, plastic free energy is related to the evolution of material micro-structure, which is not represented by specific parameters (like the internal variables in some elastoplasticity models). According
to the experimental behavior of the material models, a few assumptions that ensures their energy behavior to follow the proposed thermomechanical framework were made. Equations for energy computation were derived and implemented in Real-ESSI, which was used to perform numerical simulations in this study.

Presented approach was illustrated and tested using several column and frame models with different loading conditions. As expected, energy balance was maintained during entire simulation in all tested cases. It was shown that plastic work could drop but plastic dissipation always maintained nonnegative during any time period, which is expressed in the second law of thermodynamics. It was also observed that the difference between plastic work and plastic dissipation could be significant in most cases. The ratio between them (Quinney–Taylor coefficient) evolved with time and thus should not be assumed to be constant in general.

Figure 104.103: Energy analysis of steel frame model under imposed seismic motion.
104.13 Localization of Deformation

(Rudnicki and Rice, 1975), (Lu et al., 2009)
Chapter 105

Probabilistic Elasto-Plasticity and Stochastic Elastic-Plastic Finite Element Method

(In collaboration with Prof. Kallol Sett, and Dr. Hexiang Wang)
105.1 Chapter Summary and Highlights

For more details on work in this area, please consult the following papers:

Jeremić et al. (2007a),
Sett et al. (2007a),
Sett et al. (2007b),
Jeremić and Sett (2007),
Jeremić and Sett (2006),
Sett and Jeremić (2007),
Jeremić and Sett (2009a),
Sett and Jeremić (2010),
Sett et al. (2011b),
Sett et al. (2011a),
Jeremić and Sett (2010).

Material from (some of) the above papers is presented below.

105.2 Probabilistic Elasto-Plasticity

A second-order exact expression for evolution of Probability Density Function (PDF) of stress is derived for general, one dimensional (1-D) elastic-plastic constitutive rate equations with uncertain material parameters. The Eulerian–Lagrangian (EL) form of Fokker–Planck–Kolmogorov (FPK) equation is used for this purpose. It is also shown that by using EL form of FPK, the so called "closure problem" associated with regular perturbation methods used so far, is resolved too. The use of EL form of FPK also replaces repetitive and computationally expensive deterministic elastic-plastic computations associated with Monte Carlo technique.

The derived general expression are specialized to the particular cases of point location scale linear elastic and elastic–plastic constitutive equations, related to associated Drucker-Prager with linear hardening.

In a companion paper, the solution of FPK equations for 1D is presented, discussed and illustrated through a number of examples.

105.2.1 Probabilistic Elasto-Plasticity: Introduction

Advanced elasto–plasticity constitutive models, when properly calibrated, are very accurate in capturing important aspects of material behavior. However, all materials', and in particular geomaterials' (soil, rock,
concrete, powder, bone etc.) behavior is uncertain due to inherent spatial and point-wise uncertainties. These uncertainties in material properties could outweigh the advantages gained by using advanced constitutive models. For example, Fig. 105.1 shows a schematic of anticipated influence of material uncertainties on a bi-linear elastic-plastic stress-strain behavior. Depending on uncertainties in material properties and interaction between them, the behavior of the same material could be very different.

![Figure 105.1: Anticipated Influence of Material Fluctuations on Stress-Strain Behavior](image)

The uncertainties in material properties are inevitable in real materials and it is best to account for them in modeling and simulation. In traditional deterministic constitutive modeling, material models are calibrated against set of experimental data. Although those experimental data sets generally exhibit statistical distribution, the models are usually calibrated against the mean of the data and all the information about uncertainties is neglected.

The modeling and simulation of solids and structures with uncertain material properties involves two steps: (a) classification and quantification of uncertainties and (b) propagation of uncertainties through governing differential equations.

The uncertainties can be broadly classified into aleatory and epistemic types. Aleatory uncertainties are associated with the inherent variabilities of nature. This type of uncertainty can not be reduced. Highly developed mathematical theory is available for dealing with aleatory uncertainty. On the other hand, epistemic uncertainties arise due to our lack of knowledge. This type of uncertainty can be reduced by collecting more data but the mathematical tools to deal with them are not highly developed.
(e.g. fuzzy logic Zadeh (1983), convex models Ben-Haim and Elishakoff (1990), interval arithmetic Moore (1979) etc.). Hence, it proves useful to trade epistemic uncertainties for aleatory uncertainties in order to facilitate their propagation through the governing equations using advanced mathematical tools. It is important to note that in trading-off epistemic uncertainties for aleatory uncertainties, one doesn’t reduce the total uncertainties in the system, but assumes that the uncertainties in the system are irreducible. Under the framework of probability theory, uncertain material parameters are modeled as random variables or random fields (Vanmarcke, 1983) depending on whether they are specialized to a fixed location in their continuum or a function of location in their continuum. We note recent works in quantifying the uncertainties in material (soil) properties for geotechnical engineering applications, Lumb (1966), Vanmarcke (1977), Mayerhoff (1993), DeGroot and Baecher (1993), Popescu (1995), Lacasse and Nadim (1996), Popescu et al. (1998), Phoon and Kulhawy (1999a,b), Fenton (1999a,b), Duncan (2000b), Rackwitz (2000), Marosi and Hiltunen (2004), and Stokoe II et al. (2004) The issue of uncertain material properties becomes very pronounced when one starts dealing with the boundary value problems with uncertain material properties (elastic or elastic–plastic).

In mechanics, the equilibrium equation, \( A\sigma = \phi(t) \), together with the strain compatibility equation, \( Bu = \epsilon \), and the constitutive equation, \( \sigma = D\epsilon \), are sufficient\(^1\) to describe the behavior of the solid. Rigorous mathematical theory has been developed for problems where the only random parameter is the external force \( \phi(t) \). In this case, the probability distribution function (PDF) of the response variable will satisfy FPK partial differential equation (Soize, 1994). With appropriate initial and boundary conditions the FPK PDE can be solved for PDF of response variable. The numerical solution method for FPK equation by finite element method (FEM) is described by number of researchers e.g. Langtangen (1991), Masud and Bergman (2005).

The other extreme case, which is of interest in this work, is when the stochasticity of the system is purely due to operator uncertainty. Exact solution of the problems with stochastic operator was attempted by Hopf (1952) using characteristic functional approach. Later, Lee (1974) applied the methodology to the problem of wave propagation in random media and derived a FPK equation satisfied by the characteristic functional of the random wave field. This characteristic functional approach is very complicated for linear problems and becomes even more intractable (and possibly unsolvable) for nonlinear problems and problems with irregular geometries and boundary conditions.

Monte Carlo simulation technique is an alternative to analytical solution of partial differential equation with stochastic coefficient. Nice descriptions of different aspects of formulation of Monte Carlo technique for stochastic mechanics problem is described by Schüßler (1997). Monte Carlo method is very popular.

\(^1\)Generalized stress is \( \sigma \), \( \phi(t) \) is generalized forces that can be time dependent, \( u \) is generalized displacements, \( \epsilon \) is generalized strain and \( A, B, \) and \( D \) are operators which could be linear or non-linear.
tool with the advantage that accurate solution can be obtained for any problem whose deterministic solution (either analytical or numerical) is known. Monte Carlo technique has been used by a number of researchers in obtaining probabilistic solution of geotechnical boundary value problems, e.g. Paice et al. (1996); Griffiths et al. (2002); Fenton and Griffiths (2003, 2005). Popescu et al. (1997), Mellah et al. (2000), De Lima et al. (2001), Koutsourelakis et al. (2002), Nobahar (2003). The major disadvantage of Monte Carlo analysis is the repetitive use of the deterministic model until the solution variable become statistically significant. The computational cost associated with it could be very high especially for non–linear problems with multiple uncertain material properties.

Various difficulties in finding analytical solutions and the high computational cost associated with Monte Carlo technique instigated development of numerical method for the solution of stochastic differential equation with random coefficient. For stochastic boundary value problems Stochastic Finite Element Method (SFEM) is the most popular such method. There exist several formulations of SFEM, among which perturbation (Kleiber and Hien (1992); Der Kiureghian and Ke (1988); Mellah et al. (2000); Gutierrez and De Borst (1999)) and Spectral (Ghanem and Spanos (1991); Keese and Matthies (2002); Xiu and Karniadakis (2003); Debusschere et al. (2003); Anders and Hori (2000)) methods are very popular. A nice review on advantages and disadvantages of different formulations of SFEM was provided by Matthies et al. (1997). Mathematical issues regarding different formulations of SFEM was addressed by Deb et al. (2001) and Babuska and Chatzipantelidis (2002). It is important to note that most of the formulations described in the above mentioned references are for linear elastic problems.

A limited number of references is also available related to geometric non–linear problems, Liu and Der Kiureghian (1991); Keese and Matthies (2002) and Keese (2003)). Similarly, there exist only few published references related to material non–linear (elastic–plastic) problems with uncertain material parameters. The major difficulty in extending the available formulations of SFEM to general elastic–plastic problem is the high non–linear coupling in the elastic–plastic constitutive rate equation. First attempt to propagate uncertainties through elastic–plastic constitutive equations considering random Young’s modulus was published only recently, e.g. Anders and Hori (1999, 2000). The perturbation expansion at the stochastic mean behavior (considering only the first term of the expansion) was used in the above mentioned references. In computing the mean behavior the Authors took the advantage of bounding media approximation. Although this method doesn’t suffer from computational difficulty associated with Monte Carlo method for problems having no closed-form solution, it inherits “closure problem” and the ”small coefficient of variation” requirements for the material parameters. Closure problem refers to the need for higher order statistical moments in order to calculate lower order statistical moments Kavvas (2003). The small COV requirement claims that the perturbation method can be used (with reasonable accuracy) for probabilistic simulations of solids and structures with uncertain properties.
only if their COV < 20% (Sudret and Der Kiureghian, 2000). For soils and other natural materials, COVs are rarely below 20% (Lacasse and Nadim (1996); Phoon and Kulhawy (1999a,b)). Furthermore, with bounding media approximation, difficulty arises in computing the mean behavior when one considers uncertainties in internal variable(s) and/or direction(s) of evolution of internal variable(s).

The focus of present work is on development of methodology for the probabilistic simulation of constitutive behavior of elastic–plastic materials with uncertain properties. Recently, Kavvas (2003) obtained a generic Eulerian–Lagrangian (EL) form of FPK equation, exact to second-order, corresponding to any non-linear ordinary differential equation with random coefficients and random forcing. The approach using EL form of the FPK equation doesn’t suffer from the drawbacks of Monte Carlo method and perturbation technique. In this paper the authors applied developed EL form of the FPK equation to obtain probabilistic formulation for a general, one-dimensional incremental elastic–plastic constitutive equation with random coefficient. The solution methodology is designed with several applications in mind, namely to

- obtain probabilistic stress–strain behavior from spatial average form (upscaled form) of constitutive equation, when input uncertain material properties to the constitutive equation are random fields; and

- obtain probabilistic stress-strain behavior from point-location scale constitutive equation, when input uncertain material properties to the constitutive equation are random variables.

Application of the developed methodology is demonstrated on a particular point-location scale one-dimensional constitutive equation, namely Drucker–Prager associative linear hardening elastic–plastic material model. In this paper, derivation is made of the EL form of FPK equation that govern the 1D probabilistic elastic–plastic material models with uncertain material parameters. This general formulation is then specialized to a particular 1D Drucker–Prager associative linear hardening material model. In the companion paper the solution methodology of the FPK equation corresponding to Drucker–Prager associative linear hardening material model is described, along with illustrative examples. The methodology is general enough that it allows extension to three-dimensions and incorporation into a general stochastic finite element framework. This work is underway and will be reported in future publications.

### 105.2.2 Probabilistic Elasto-Plasticity: General Formulation

The incremental form of spatial-average elastic-plastic constitutive equation can be written as

\[
\frac{d\sigma_{ij}(x,t)}{dt} = D_{ijkl}(x,t)\frac{d\epsilon_{kl}(x,t)}{dt} \quad (105.1)
\]
where the continuum stiffness tensor \( D_{ijkl}(x, t) \) can be either elastic or elastic-plastic

\[
D_{ijkl} = \begin{cases} 
D_{el}^{ijkl} & ; f < 0 \lor (f = 0 \land df < 0) \\
D_{el}^{ijkl} - \frac{\partial f}{\partial \sigma_{rs}} D_{ijkl}^{el} - \frac{\partial f}{\partial q_r} D_{ijkl}^{el} & ; f = 0 \lor df = 0
\end{cases}
\tag{105.2}
\]

and where \( D_{el}^{ijkl} \) is the elastic stiffness tensor, \( D_{ep}^{ijkl} \) is the elastic–plastic continuum stiffness tensor, \( f \) is the yield function, which is a function of stress \( \sigma_{ij} \) and internal variables \( q_* \), \( U \) is the plastic potential function (also a function of stress and internal variables). The internal variables \( q_* \) could be scalar(s) (for perfectly-plastic and isotropic hardening models), second-order tensor (for translational and rotational kinematic hardening) or fourth-order tensor (for distortional hardening). Therefore, the most general form of incremental constitutive equation in terms of its parameters can be written as

\[
\frac{d\sigma_{ij}(x, t)}{dt} = \beta_{ijkl}(\sigma_{ij}, D_{ijkl}, q_*, r_*; x, t) \frac{d\epsilon_{kl}(x, t)}{dt} \tag{105.3}
\]

Due to randomnesses in elastic constants \( (D_{el}^{ijkl}) \) and internal variables \( (q_*) \) and/or rate of evolution of internal variables \( (r_*) \) the material stiffness operator \( \beta_{ijkl} \) in Eq. (105.3) becomes stochastic. It follows that the Equation (105.1) becomes a linear/non-linear ordinary differential equations with stochastic coefficients. Similarly, randomness in he forcing term \( (\epsilon_{kl}) \) of Equation (105.3) results in Equation (105.3) becoming linear/non-linear ordinary differential equations with stochastic forcing. This can be generalized, so that randomnesses in material properties and forcing function of Equation (105.3) results in Equation (105.3) becoming a linear/non-linear ordinary differential equation with stochastic coefficients and stochastic forcing.

In order to gain better understanding of the effects of random material parameters and forcing on response, focus is shifted from a general 3D case to a 1D case. In what follows, the probabilistic formulation for 1-D constitutive elastic–plastic incremental equation with stochastic coefficient and stochastic forcing is derived. In addition to that, the probabilistic formulation for 1-D elastic linear constitutive equation is obtained as a special case of non-linear general derivation.

Focusing on 1-D behavior, the Eq. (105.3) is written as

\[
\frac{d\sigma(x, t)}{dt} = \beta(\sigma, D, q, r; x, t) \frac{d\epsilon(x, t)}{dt} \tag{105.4}
\]

which is a non-linear ordinary differential equation with stochastic coefficient and stochastic forcing. The right hand side of Eq. (105.4) is replaced with the function \( \eta \) as

\[
\eta(\sigma, D, q, r, c; x, t) = \beta(\sigma, D, q, r; x, t) \frac{d\epsilon(x, t)}{dt} \tag{105.5}
\]
so that now Eq. (105.4) can be written as
\[
\frac{\partial \sigma(x_t, t)}{\partial t} = \eta(\sigma, D, q, r, \epsilon; x, t) \tag{105.6}
\]
with initial condition,
\[
\sigma(x, 0) = \sigma_0 \tag{105.7}
\]

In the above Eq. (105.6) \(\sigma\) can be considered to represent a point in the \(\sigma\)-space and hence, the Eq. (105.6) determines the velocity for the point in that \(\sigma\)-space. This may be visualized, from the initial point, and given initial condition \(\sigma_0\), as a trajectory that describes the corresponding solution of the non-linear stochastic ordinary differential equation (ODE) (Eq. (105.6)). Considering now a cloud of initial points (refer to Fig. 105.2), described by a density \(\rho(\sigma_0)\) in the \(\sigma\)-space.

![Figure 105.2: Movements of Cloud of Initial Points, described by density \(\rho(\sigma, 0)\), in the \(\sigma\)-space](image)

The phase density \(\rho\) of \(\sigma(x, t)\) (movement of any point dictated by Eq. (105.6)) varies in time according to a continuity equation which expresses the conservation of all these points in the \(\sigma\)-space. This continuity equation can be expressed in mathematical terms, using Kubo’s stochastic Liouville equation (Kubo, 1963):

\[
\frac{\partial \rho(\sigma(x, t), t)}{\partial t} = -\frac{\partial}{\partial \sigma} \eta[\sigma(x, t), D(x), q(x), r(x), \epsilon(x, t)], \rho[\sigma(x, t), t] \tag{105.8}
\]

with an initial condition,
\[
\rho(\sigma, 0) = \delta(\sigma - \sigma_0) \tag{105.9}
\]
where \( \delta \) is the Dirac delta function and Eq. (105.9) is the probabilistic restatement in the \( \sigma \)-phase space of the original deterministic initial condition (Eq. (105.7)). Here it proves useful to recall Van Kampen’s Lemma (Van Kampen, 1976), which states that the ensemble average of a phase density is the probability density

\[
< \rho(\sigma, t) > = P(\sigma, t)
\]

(105.10)

where, the symbol \( < \cdot > \) denotes the expectation operation, and \( P(\sigma, t) \) denotes evolutionary probability density of the state variable \( \sigma \) of the constitutive rate equation (Eq. (105.4)).

In order to obtain the deterministic probability density function (PDF) \( (\sigma, t) \) of the state variable, it is necessary to obtain the deterministic partial differential equation (PDE) of the \( \sigma \)-space mean phase density \( < \rho(\sigma, t) > \) from the linear stochastic PDE system (Eqs. (105.8) and (105.9)). This necessitates the derivation of the ensemble average form of Eq. (105.8) for \( < \rho(\sigma, t) > \). This ensemble average was recently derived by (Kavvas and Karakas, 1996; Kavvas, 2003) as

\[
\frac{\partial}{\partial t} \left< \rho(\sigma(x_t, t), t) \right> = \\
- \frac{\partial}{\partial \sigma} \left\{ \left< \eta(\sigma(x_t, t), D(x_t), q(x_t), r(x_t), \epsilon(x_t, t)) \right> \right\} \\
- \int_0^t \frac{d\tau}{d\sigma} \text{Cov}_0 \left[ \left< \eta(\sigma(x_t, t - \tau), D(x_{t-\tau}), q(x_{t-\tau}), r(x_{t-\tau}), \epsilon(x_{t-\tau}, t - \tau)) \right> \right] \left< \rho(\sigma(x_t, t), t) \right> \right\} \\
+ \frac{\partial}{\partial \sigma} \left\{ \int_0^t \text{Cov}_0 \left[ \left< \eta(\sigma(x_{t-\tau}, t - \tau), D(x_{t-\tau}), q(x_{t-\tau}), r(x_{t-\tau}), \epsilon(x_{t-\tau}, t - \tau)) \right> \right] \frac{\partial \left< \rho(\sigma(x_t, t), t) \right>}{\partial \sigma} \right\}
\]

(105.11)

to exact second order (to the order of the covariance time of \( \eta \)). In Eq. (105.11), \( \text{Cov}_0[\cdot] \) is the time ordered covariance function defined by

\[
\text{Cov}_0 \left[ \eta(x, t_1), \eta(x, t_2) \right] = \langle \eta(x, t_1) \eta(x, t_2) \rangle - \langle \eta(x, t_1) \rangle \cdot \langle \eta(x, t_2) \rangle
\]

(105.12)

By combining Eqs. (105.11) and (105.10) and rearranging the terms yields the following Fokker–Planck equation (FPE, also known as Forward–Kolmogorov Equation or Fokker–Planck–Kolmogorov...
FPK Equation) (Risken (1989), Gardiner (2004), Schüller (1997)):

\[
\frac{\partial P(\sigma(x,t),t)}{\partial t} = \\
- \frac{\partial}{\partial \sigma} \left\{ \left\langle \eta(\sigma(x,t), D(x_t), q(x_t), r(x_t)) \right\rangle \right\} \\
+ \int_0^t d\tau \text{Cov}_0 \left[ \frac{\partial \eta(\sigma(x_t, t), D(x_t), q(x_t), r(x_t))}{\partial \sigma}; \eta(\sigma(x_{t-\tau}, t-\tau), D(x_{t-\tau}), q(x_{t-\tau}), r(x_{t-\tau}), \epsilon(x_{t-\tau}, t-\tau)) \right] P(\sigma(x,t), t) \\
+ \frac{\partial^2}{\partial \sigma^2} \left\{ \left\langle \int_0^t d\tau \text{Cov}_0 \left[ \eta(\sigma(x_t, t), D(x_t), q(x_t), r(x_t), \epsilon(x_t)) \right]; \eta_1(\sigma(x_{t-\tau}, t-\tau), D(x_{t-\tau}), q(x_{t-\tau}), r(x_{t-\tau}), \epsilon(x_{t-\tau}, t-\tau)) \right] \right\} P(\sigma(x,t), t) \\
\]  

(105.13)

to exact second order. This is the most general relation for probabilistic behavior of inelastic (non–linear, elastic–plastic) 1-D stochastic incremental constitutive equation. The solution of this deterministic linear FPE (Eq. (105.13)), in terms of the probability density \( P(\sigma, t) \), under appropriate initial and boundary conditions will yield the PDF of the state variable \( \sigma \) of the original 1-D non-linear stochastic constitutive rate equation (Eq. (105.4)). It is important to note that while the original equation (Eq. (105.4)) is non-linear, the FPE (Eq. (105.13)) is linear in terms of its unknown, the probability density \( P(\sigma, t) \) of the state variable \( \sigma \). This linearity, in turn, provides significant advantages in the solution of the probabilistic behavior of the incremental constitutive equation (Eq. (105.4)).

One should also note that Eq. (105.13) is a mixed Eulerian-Lagrangian equation. This stems from the fact that while the real space location \( x_t \) at time \( t \) is known, the location \( x_{t-\tau} \) is an unknown. If one assumes small strain theory, one can relate the unknown location \( x_{t-\tau} \) from the known location \( x_t \) by using the strain rate, \( \dot{\epsilon} (=d\varepsilon/dt) \) as,

\[
x_{t-\tau} = (1 - \dot{\varepsilon}\tau)x_t
\]  

(105.14)

Once the probability density function \( P(\sigma, t) \) is obtained it can be used to obtain the mean of state variable \( \langle \sigma \rangle \) by usual expectation operation

\[
\langle \sigma(t) \rangle = \int \sigma(t) P(\sigma(t)) d\sigma(t)
\]  

(105.15)

Another possible way to obtain the mean of state variable is to use the equivalence between FPE and Itô stochastic differential equation (Gardiner, 2004). In this case Itô stochastic differential equation
equivalent to Eq. (105.13) is

\[
\begin{align*}
d\sigma(x,t) &= \left\{ \eta(\sigma(x,t), D(x_t), q(x_t), r(x_t), \epsilon(x_t, t)) \right. \\
& \quad + \int_0^t d\tau Cov_0 \left[ \frac{\partial\eta(\sigma(x,t), D(x_t), q(x_t), r(x_t), \epsilon(x_t, t))}{\partial \sigma} \right. \\
& \quad \left. \eta(\sigma(x_{t-\tau}, t-\tau), D(x_{t-\tau}), q(x_{t-\tau}), r(x_{t-\tau}), \epsilon(x_{t-\tau}, t-\tau)) \right) dt \\
& \quad + b(\sigma, t)dW(t) \\
\end{align*}
\]

(105.16)

where,

\[
\begin{align*}
b^2(\sigma, t) &= 2 \int_0^t d\tau Cov_0 \left[ \eta(\sigma(x,t), D(x_t), q(x_t), r(x_t), \epsilon(x_t, t)); \\
& \quad \eta(\sigma(x_{t-\tau}, t-\tau), D(x_{t-\tau}), q(x_{t-\tau}), r(x_{t-\tau}), \epsilon(x_{t-\tau}, t-\tau)) \right]
\end{align*}
\]

(105.17)

and, \(dW(t)\) is an increment of Wiener process \(W\) with \(<dW(t)> = 0\). It is also interesting to note that all the stochasticity of the original equation (Eq. (105.4)) are lumped together in the last term (Wiener increment term) of the right-hand-side of Eq. (105.16). By taking advantage of the independent increment property of the Wiener process \(<dW(t)> = 0\), one can derive the differential equation which describes the evolution of mean of state variable \((\sigma)\) of the nonlinear constitutive rate equation in time and space as, (e.g. (Kavvas, 2003))

\[
\begin{align*}
\frac{<da(x,t)>}{dt} &= \left\{ \eta(\sigma(x,t), D(x_t), q(x_t), r(x_t), \epsilon(x_t, t)) \right. \\
& \quad + \int_0^t d\tau Cov_0 \left[ \frac{\partial\eta(\sigma(x,t), D(x_t), q(x_t), r(x_t), \epsilon(x_t, t))}{\partial \sigma} \right. \\
& \quad \left. \eta(\sigma(x_{t-\tau}, t-\tau), D(x_{t-\tau}), q(x_{t-\tau}), r(x_{t-\tau}), \epsilon(x_{t-\tau}, t-\tau)) \right]
\end{align*}
\]

(105.18)

Eq. (105.18) is a nonlocal integro-differential equation in Eulerian-Lagrangian form, since, although the location \(x_t\) at time \(t\) is known, the Lagrangian location \(x_{t-\tau}\) is an unknown which is determined by Eq. (105.14). It is important to note that the state variable appearing within \(\eta(\cdot)\) on the right-hand-side of Eq. (105.18) is random and needs to be treated accordingly.

This concludes the development of relation for probabilistic behavior of 1-D elastic–plastic constitutive incremental equation with stochastic coefficients and stochastic forcing in most general form. In the following section the developed general relation is specialized to two particular types of point-location scale constitutive modeling: a) 1-D (shear) linear elastic constitutive behavior, and b) 1-D (shear) elastic-plastic Drucker-Prager associative linear hardening constitutive behavior.
105.2.3 Probabilistic Elasto-Plasticity: Elastic–Plastic Probabilistic 1-D Constitutive Incremental Equation

For materials obeying Drucker-Prager yield criteria (without cohesion), the yield surface can be written as:

\[ f = \sqrt{J_2} - \alpha I_1 \]  

(105.19)

where \( J_2 = \frac{1}{2} s_{ij} s_{ij} \) is the second invariant of the deviatoric stress tensor \( s_{ij} = \sigma_{ij} - \frac{1}{3} \delta_{ij} \sigma_{kk} \), and \( I_1 = \sigma_{ii} \) is the first invariant of the stress tensor, and \( \alpha \), an internal variable, is a function of friction angle \( \alpha = \frac{2 \sin(\phi)}{\sqrt{3}(3 - \sin(\phi))} \), where \( \phi \) is the friction angle (e.g. (Chen and Han, 1988b)).

By assuming associative flow rule, so that the yield function has the same derivatives as the plastic flow function

\[ \frac{\partial f}{\partial \sigma_{ij}} = \frac{\partial U}{\partial \sigma_{ij}} \]  

(105.20)

one can expand parts of the tangent constitutive tensor given in Eq. 105.2 (from Page 404), to read \(^2\)

\[ A_{kl} = \frac{\partial f}{\partial \sigma_{pq}} D_{pqkl} = A_{kl} \frac{\partial f}{\partial I_1} \left( 2G \left( \frac{\partial I_1}{\partial \sigma_{11}} \delta_{11} \delta_{lk} + \frac{\partial I_1}{\partial \sigma_{22}} \delta_{22} \delta_{lk} + \frac{\partial I_1}{\partial \sigma_{33}} \delta_{33} \delta_{lk} \right) \right. \]

\[ + \left. \left( K - \frac{2}{3} G \right) \frac{\partial I_1}{\partial \sigma_{cd}} \delta_{cd} \delta_{lk} \right) \]

\[ + \frac{\partial f}{\partial \sqrt{J_2}} \left( 2G \frac{\partial \sqrt{J_2}}{\partial \sigma_{ij}} \delta_{ik} \delta_{jl} + \left( K - \frac{2}{3} G \right) \frac{\partial \sqrt{J_2}}{\partial \sigma_{ab}} \delta_{ab} \delta_{kl} \right) \]  

(105.21)

and,

\[ B = \frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}} = \left( \frac{\partial f}{\partial I_1} \right)^2 \left( 2G \left( \left( \frac{\partial I_1}{\partial \sigma_{11}} \right)^2 + \left( \frac{\partial I_1}{\partial \sigma_{22}} \right)^2 + \left( \frac{\partial I_1}{\partial \sigma_{33}} \right)^2 \right) \right) \]

\[ + \left( K - \frac{2}{3} G \right) \left( \frac{\partial I_1}{\partial \sigma_{ij}} \delta_{ij} \right)^2 \]

\[ + \left( \frac{\partial f}{\partial \sqrt{J_2}} \right)^2 \left( 2G \frac{\partial \sqrt{J_2}}{\partial \sigma_{ij}} \frac{\partial \sqrt{J_2}}{\partial \sigma_{ij}} + \left( K - \frac{2}{3} G \right) \left( \frac{\partial \sqrt{J_2}}{\partial \sigma_{ij}} \delta_{ij} \right)^2 \right) \]  

(105.22)

where, \( K \) and \( G \) are the elastic bulk modulus and the elastic shear modulus respectively.

\(^2\)A more detailed derivation of this probabilistic differentiation is given in the Appendix.
By further assuming that the evolution of internal variable is a function of equivalent plastic strain, $e_{eq}^p = 2/3 e_{ij}^p e_{ij}^p$ then one can write
\[
K_P = -\frac{\partial f}{\partial q_n} r_n = -\frac{1}{\sqrt{3}} \frac{\partial f}{\partial e_{eq}^p} \frac{d\alpha}{d\sqrt{J_2} \partial f} \quad (105.23)
\]

It should be noted that since material properties are assumed to be random, the resulting stress tensor will also become random and hence the derivatives of the stress invariants with respect to stress tensor ($\sigma_{ij}$) will become random. Therefore, differentiations appearing in Eqs. (105.21), (105.22), and (105.23) can not be carried out in a deterministic sense.

The parameter tensor in Eq. (105.1) then becomes
\[
D^p_{ijkl} = \begin{cases} 
2G\delta_{ik}\delta_{jl} + \left(K - \frac{2}{3}G\right)\delta_{ij}\delta_{kl} & ; f < 0 \lor (f = 0 \land df < 0) \\
2G\delta_{ik}\delta_{jl} + \left(K - \frac{2}{3}G\right)\delta_{ij}\delta_{kl} - \frac{A_{ij}A_{kl}}{B + K_P} & ; f = 0 \lor df = 0 
\end{cases} \quad (105.24)
\]

where tensor $A_{ij}$ and scalars $B$ and $K_P$ are defined by Eqs. (105.21), (105.22), and (105.23) respectively. The above equation (Eq. 105.24) represents a probabilistic continuum stiffness tensor for an elastic–plastic material model, in this case Drucker-Prager isotropic linear hardening material with associated plasticity. By focusing our attention on one dimensional point-location scale shear constitutive relationship between $\sigma_{12}$ and $\epsilon_{12}$ for Drucker-Prager material model, one can simplify the function $\eta(\sigma, D, q, r, \epsilon; x, t)$ as defined in Eq. (105.5) (on Page 404) to read
\[
\eta = \begin{cases} 
2G \frac{d\epsilon_{12}}{dt} & ; f < 0 \lor (f = 0 \land df < 0) \\
\left(\frac{4G^2}{2G - \frac{4G^2}{B + K_P} \left(\frac{\partial f}{\sqrt{J_2} \partial \sigma_{12}}\right)^2}\right) \frac{d\epsilon_{12}}{dt} & ; f = 0 \lor df = 0 
\end{cases} \quad (105.25)
\]

By considering both the material properties (shear modulus $G$, bulk modulus $K$, friction angle $\alpha$, and rate of change of friction angle (linear hardening) $\alpha'$) and the strain rate ($d\epsilon_{12}/dt(t)$) as random, one can substitute $\eta$ as derived in Eq. (105.13) to obtain the particular FPK equation for the probabilistic behavior of Drucker-Prager associative linear hardening, 1-D point-location scale elastic-plastic shear constitutive rate equation. In particular, two cases are recognized, one for elastic (pre–yield) behavior of

---

3This is a fairly common assumption, e.g. (Chen and Han, 1988b)
material \((f < 0 \lor (f = 0 \land df < 0))\)

\[
\frac{\partial P(\sigma_{12}(t), t)}{\partial t} = \\
- \frac{\partial}{\partial \sigma_{12}} \left[ \left\langle 2G \frac{d\epsilon_{12}^e}{dt}(t) \right\rangle P(\sigma_{12}(t), t) \right] \\
+ \frac{\partial^2}{\partial \sigma_{12}^2} \left[ \left\{ \int_0^t d\tau \text{Cov}\left[ 2G \frac{d\epsilon_{12}^e}{dt}(t) ; 2G \frac{d\epsilon_{12}^e}{dt}(t - \tau) \right] \right\} P(\sigma_{12}(t), t) \right] \\
+ \frac{\partial^2}{\partial \sigma_{12}^2} \left[ \left\{ \int_0^t d\tau \text{Cov}\left[ G_{ep}(t) \frac{d\epsilon_{12}^e}{dt}(t) ; G_{ep}(t - \tau) \frac{d\epsilon_{12}^e}{dt}(t - \tau) \right] \right\} P(\sigma_{12}(t), t) \right] \\
(105.26)
\]

noting that this is the same equation as Eq. (105.3). In addition to that, the case of elastic–plastic behavior \((f = 0 \lor df = 0)\) is described by the following probabilistic equation

\[
\frac{\partial P(\sigma_{12}(t), t)}{\partial t} = - \frac{\partial}{\partial \sigma_{12}} \left[ \left\langle G_{ep}(t) \frac{d\epsilon_{12}^e}{dt}(t) \right\rangle \right] \\
+ \int_0^t d\tau \text{Cov}\left[ G_{ep}(t) \frac{d\epsilon_{12}^e}{dt}(t) ; G_{ep}(t - \tau) \frac{d\epsilon_{12}^e}{dt}(t - \tau) \right] P(\sigma_{12}(t), t) \right] \\
+ \frac{\partial^2}{\partial \sigma_{12}^2} \left[ \left\{ \int_0^t d\tau \text{Cov}\left[ G_{ep}(t) \frac{d\epsilon_{12}^e}{dt}(t) ; G_{ep}(t - \tau) \frac{d\epsilon_{12}^e}{dt}(t - \tau) \right] \right\} P(\sigma_{12}(t), t) \right] \\
(105.27)
\]

where \(G_{ep}(a)\) is defined as probabilistic elastic–plastic kernel and is introduced to shorten the writing (but will also have other uses later)

\[
G_{ep}(a) = \left( 2G - \frac{4G^2 \left( \frac{\partial f}{\partial \sqrt{J_2(a)}} \frac{\partial \sqrt{J_2(a)}}{\partial \sigma_{12}(a)} \right)^2}{B(a) + K_P(a)} \right) \\
(105.28)
\]

and \(a\) assumes values \(t\) or \(t - \tau\).

It is important to note that the differentiations appearing in the coefficient terms of the FPK PDE (Eq. (105.27)), within the probabilistic elastic–plastic kernel \(G_{ep}(a)\) (i.e. Eq. (105.28)), are for fixed values of \(\sigma_{12}\) and hence those differentiations can be carried out in a deterministic sense. After carrying out the differentiations, the probabilistic elastic–plastic kernel becomes

\[
G_{ep}(a)\big|_{\sigma_{12} \rightarrow \text{const.}} = \left( 2G - \frac{G^2}{G + 9K \alpha^2 + \frac{1}{\sqrt{d}} I_1(a) \alpha} \right) \\
(105.29)
\]

which, after substitution, result in simplification of the FPK equation (105.27). Further simplification is possible by noting that the first random process in the covariance term of the first coefficient on the r.h.s of the equation (105.27) is independent of \(\sigma_{12}\). Furthermore, since the covariance of zero with any
random process is zero, the FPK equation (105.27) is further simplified to read
\[
\frac{\partial P}{\partial t}(\sigma_{12}(t), t) = -\frac{\partial}{\partial \sigma_{12}} \left[ \left\langle G^{ep}(t) \frac{d\sigma_{12}}{dt}(t) \right\rangle P(\sigma_{12}(t), t) \right] \\
+ \frac{\partial^2}{\partial \sigma_{12}^2} \left\{ \int_0^t d\tau \text{Cov}_0 \left[ \left\langle G^{ep}(t) \frac{d\sigma_{12}}{dt}(t); G^{ep}(t-\tau) \frac{d\sigma_{12}}{dt}(t-\tau) \right\rangle \right] P(\sigma_{12}(t), t) \right\}
\]
(105.30)

where the probabilistic elastic–plastic kernel \(G^{ep}(a)\) is given by the Eq. (105.29).

The evolution of a mean value of shear stress \(\sigma_{12}\) is obtained by substituting \(\eta\) (derived for Drucker-Prager material in Eq. (105.18)) as
\[
< \frac{d\sigma_{12}(t)}{dt} > = \left\langle G^{ep}(t) \frac{d\sigma_{12}}{dt}(t) \right\rangle \\
+ \int_0^t d\tau \text{Cov}_0 \left[ \left( \frac{\partial}{\partial \sigma_{12}} \left\langle G^{ep}(t) \frac{d\sigma_{12}}{dt}(t); G^{ep}(t-\tau) \frac{d\sigma_{12}}{dt}(t-\tau) \right\rangle \right) \right] P(\sigma_{12}(t), t)
\]
(105.31)

It is important to note that the derivatives appearing in the mean and covariance term of the above Eulerian-Lagrangian integro-differential equation (Eq. (105.31) with the probabilistic elastic–plastic kernel defined through the Eq. (105.29)) are random differentiations and need to be treated accordingly. One possible approach to obtaining these differentiations could be perturbation with respect to mean (Anders and Hori, 2000) but the "closure problem" will appear. Hence, in this study the evolution of mean of \(\sigma_{12}\) will be obtained by the expectation operation on the PDF (Eq. (105.15)).

### 105.2.4 Probabilistic Elasto-Plasticity: Initial and Boundary Conditions for the Probabilistic Elastic–Plastic PDE

The PDE describing the probabilistic behavior of constitutive rate equations can be written in the following general form:
\[
\frac{\partial P(\sigma_{12}, t)}{\partial t} = -\frac{\partial}{\partial \sigma_{12}} \left\{ P(\sigma_{12}, t)N(1) \right\} + \frac{\partial^2}{\partial \sigma_{12}^2} \left\{ P(\sigma_{12}, t)N(2) \right\}
\]
\[
= -\frac{\partial}{\partial \sigma_{12}} \left[ P(\sigma_{12}, t)N(1) - \frac{\partial}{\partial \sigma_{12}} \left\{ P(\sigma_{12}, t)N(2) \right\} \right]
\]
(105.32)

where, \(N(1)\) and \(N(2)\) are coefficients\(^4\) of the PDE and represent the expressions within the curly braces of the first and second terms respectively on the right–hand–side of Eqs. (??), (105.26), and (105.27). These terms are called the advection \((N(1))\) and diffusion \((N(2))\) coefficients as the form of Eq. (105.32)

\(^4\)Indices in brackets are not used in index summation convention.
closely resembles advection–diffusion equation (Gardiner, 2004). The symbol \( \zeta \) in Eq. (105.32) can be considered to be the probability current. This follows from Eq. (105.32), which is a continuity equation and the state variable of the equation is probability density.

After introducing initial and boundary conditions, one can solve Eq. (105.32) for probability densities of \( \sigma_{12} \) with evolution of time. The initial condition could be deterministic or stochastic depending on the type of problem. For probabilistic behavior of linear elastic constitutive rate equation (Eq. (?)), one can assume that all the probability mass at time \( t = 0 \) is concentrated at \( \sigma_{12} = 0 \) or at some constant value of \( \sigma_{12} \) if there were some initial stresses to begin with (e.g. overburden pressure on a soil mass).

In mathematical term, this translates to,

\[
P(\sigma_{12},0) = \delta(\sigma_{12})
\]

where, \( \delta(\cdot) \) is the Dirac delta function.

For the post–yield behavior of probabilistic elastic-plastic constitutive rate Equation 5 (105.30), there will be a distribution of \( \sigma_{12} \), corresponding to the solution of the pre-yield probabilistic behavior (Eq. (105.26)), to begin with. This probability mass \( P(\sigma_{12}(t),t) \), dictated by Eq. (105.13), will advect and diffuse into the domain \( (\sigma_{12},t \text{ space}) \) of the system throughout the evolution (in time/strain) of the simulation. Since it is required that the probability mass within the system is conserved i.e. no leaking is allowed at the boundaries, a reflecting barrier at the boundaries will be the preferred choice.

In mathematical term, one can express this condition as (Gardiner, 2004)

\[
\zeta(\sigma_{12},t)\big|_{\text{At Boundaries}} = 0
\]

In theory, the stress domain could extend from \( -\infty \) to \( \infty \) so that boundary conditions are then

\[
\zeta(-\infty,t) = \zeta(\infty,t) = 0
\]

With these initial and boundary conditions, the probabilistic differential equation (with random material properties and random strain) for elasto–plasticity, specialized in this case to associated Drucker–Prager material model with linear hardening, and by using FPK transform described above, can be solved for probability densities of shear stress \( (\sigma_{12}) \) as it evolves with time/shear strain \( (\epsilon_{12}) \).

### 105.2.5 Probabilistic Elasto-Plasticity: Fokker–Planck–Kolmogorov Equation for Probabilistic Elasticity and Elasto–Plasticity in 1-D

By focusing attention to the randomness of material properties only (i.e. assuming the forcing function (strain rate) as deterministic), partial differential equation (PDE) describing the evolution of probability

---

5Specialized to Drucker-Prager associated linear hardening model.
density function (PDF) of stress can be simplified. In particular, for 1D case, and for linear elastic material (but still with probabilistic material properties, in this case shear modulus $G$) one can write the following PDE
\[
\frac{\partial P(\sigma_{12}(t))}{\partial t} = - \left( 2G \frac{d\epsilon_{12}}{dt} \right) \frac{\partial P(\sigma_{12}(t))}{\partial \sigma_{12}} + \left\{ \int_{0}^{t} d\tau \text{Cov}_{0} \left[ 2G \frac{d\epsilon_{12}}{dt}; 2G \frac{d\epsilon_{12}}{dt} \right] \right\} \frac{\partial^{2} P(\sigma_{12}(t))}{\partial \sigma_{12}^{2}}
\]
(105.36)

Similarly, for elastic–plastic state, again by neglecting the randomness in strain rate, one can write the PDE for evolution of PDF of stress in 1D as
\[
\frac{\partial P(\sigma_{12}(t))}{\partial t} = - \left( G_{\text{ep}}(t) \frac{d\epsilon_{12}}{dt} \right) \frac{\partial P(\sigma_{12}(t))}{\partial \sigma_{12}} + \left\{ \int_{0}^{t} d\tau \text{Cov}_{0} \left[ G_{\text{ep}}(t) \frac{d\epsilon_{12}}{dt}; G_{\text{ep}}(t-\tau) \frac{d\epsilon_{12}}{dt} \right] \right\} \frac{\partial^{2} P(\sigma_{12}(t))}{\partial \sigma_{12}^{2}}
\]
(105.37)

where $G_{\text{ep}}(a)$ is the probabilistic elastic–plastic tangent stiffness, (given in Jeremić et al. (2007a))
\[
G_{\text{ep}}(a) = 2G - \frac{G^{2}}{G + 9K\alpha^{2} + \frac{1}{\sqrt{3}}I_{1}(a)\alpha'}
\]
(105.38)

where in the previous equation (105.38), $a$ assumes values $t$ or $t - \tau$. With appropriate initial and boundary conditions as described in Jeremić et al. (2007a), one can solve Eqs. (105.36) and (105.37) for evolution of PDF of shear stress with shear strain.

105.2.6 Probabilistic Elasto-Plasticity: Example Problem Statements

The applicability of proposed FPK equations (Eqs. (105.36) and (105.37)) in describing probabilistic elasto-plastic behavior, is verified using the following three example problems.

Problem I. Assume the material is linear elastic, probabilistic, with probabilistic shear modulus ($G$) given by a normal distribution at a point–location scale with mean of 2.5 MPa and standard deviation of 0.707 MPa. The aim is to calculated the evolution of PDF of shear stress ($\sigma_{12}$) with shear strain ($\epsilon_{12}$) for a displacement-controlled test with deterministic shear strain increment. The other parameters are considered deterministic and are as follows: Poisson’s ratio ($\nu = 0.2$, and confining pressure $I_{1} = 0.03$ MPa.

Problem II. Assume elastic–plastic material model, composed of linear elastic component and Drucker–Prager associative isotropic linear hardening elastic–plastic component. The probabilistic shear modulus ($G$) is given through a normal distribution at a point–location scale with mean of 2.5 MPa
and standard deviation of 0.707 MPa. The aim is to calculate the evolution of the PDF of shear stress \( \sigma_{12} \) with shear strain \( \epsilon_{12} \) for a displacement-controlled test with deterministic shear strain increment. The other parameters are considered deterministic and are as follows: Poisson’s ratio \( \nu = 0.2 \), confining pressure \( I_1 = 0.03 \) MPa, yield parameter\(^6\) \( \alpha = 0.071 \), plastic slope\(^7\) \( \alpha' = 5.5 \).

**Problem III.** Assume elastic–plastic material model, with linear elastic component and Drucker–Prager associative isotropic linear hardening elastic–plastic component. The probabilistic yield parameter \( \alpha \) is given through a normal distribution at a point-location scale with mean of 0.52 and standard deviation of 0.1. The aim is to calculate the evolution of the PDF of shear stress \( \sigma_{12} \) with shear strain \( \epsilon_{12} \) for a displacement-controlled test with deterministic shear strain increment. The other parameters are considered deterministic and are as follows: shear modulus \( G = 2.5 \) MPa, Poisson’s ratio \( \nu = 0.2 \), confining pressure \( I_1 = 0.03 \) MPa, and the plastic slope \( \alpha' = 5.5 \).

The above three problems will be solved using the proposed FPK equation approach. In addition to that, the solution will verified using either variable transformation method, for linear elastic case or repetitive Monte Carlo type simulations for elastic-plastic case.

### 105.2.7 Probabilistic Elasto-Plasticity: Determination of Coefficients for Fokker–Planck–Kolmogorov Equation

To solve Problems I, II, and III, the advection and diffusion coefficients \( N_1 \) and \( N_2 \) must be determined for all three problems. For sake of simplicity, a constant strain rate is assumed and hence, terms containing \( d\epsilon_{12}/dt \) in coefficients of Eqs. (105.36) and (105.37) can be substituted by a constant numerical value for the entire simulation of the evolution of PDF. It should be noted that the FPK equation (Eqs. (105.36) or (105.37)) describes the evolution of PDFs of stress with time, while, similarly, strain rate describes the evolution of strain with time. Combining the two, the evolution of PDF of stress with strain can be obtained. Time has been brought in this simulation as an intermediate dimension to help in solution process, and hence, the numerical value of strain rate could be any arbitrary value, which will cancel out once the time evolution of PDF of stress is converted to strain evolution of PDF of stress. For simulation of all the three example problems, an arbitrary value of strain rate of \( d\epsilon_{12}/dt = 0.0541/s \) is assumed.

It should also be noted that since the material properties are assumed as random variables at a point-location scale, the covariance terms appearing within the advection and diffusion coefficients become variances of random variables. For estimations of means and variances of functions of random variables (e.g. for Problems II and III) from basic random variables, commercially available statistical software

---

\(^6\)The yield parameter \( \alpha \) is an internal variable and is a function of friction angle \( \phi \) given by \( \alpha = 2 \sin(\phi) / (\sqrt{3}(3 - \sin \phi)) \) (e.g. (Chen and Han, 1988b))

\(^7\)The plastic slope \( \alpha' \) is a rate of change of friction angle governing linear hardening.
mathStatica Rose and Smith (2002) was used.

Substituting the values of deterministic and random material properties and the strain rate, coefficients $N_{(1)}$ and $N_{(2)}$ of the FPK equations can be obtained for all problems:

**Problem I**

\[
N_{(1)} = \left\langle 2G \frac{d\epsilon_{12}}{dt} \right\rangle \\
= 2 \frac{d\epsilon_{12}}{dt} \langle G \rangle \\
= 0.27 \text{ MPa/s}
\]

\[
N_{(2)} = \int_0^t \! d\tau \text{Var} \left[ 2G \frac{d\epsilon_{12}}{dt} \right] \\
= 4t \left( \frac{d\epsilon_{12}}{dt} \right)^2 \text{Var}[G] \\
= 0.0058t \text{ (MPa/s)}^2
\]

**Problem II**

For pre-yield linear elastic case, the coefficients $N_{(1)}$ and $N_{(2)}$ will be the same as those for Problem I. For post-yield elastic-plastic case the coefficients are

\[
N_{(1)} = \left\langle \left( 2G - \frac{G^2}{G + 9K\alpha^2 + \frac{1}{\sqrt{3}}I_1\alpha'} \right) \frac{d\epsilon_{12}}{dt} \right\rangle \\
= \frac{d\epsilon_{12}}{dt} \left( 2G - \frac{G^2}{G + 9K\alpha^2 + \frac{1}{\sqrt{3}}I_1\alpha'} \right) \\
= 0.147 \text{ MPa/s}
\]

\[
N_{(2)} = t \left( \frac{d\epsilon_{12}}{dt} \right)^2 \text{Var} \left[ 2G - \frac{G^2}{G + 9K\alpha^2 + \frac{1}{\sqrt{3}}I_1\alpha'} \right] \\
= 0.00074t \text{ (MPa/s)}^2
\]

**Problem III**
For post-yield elastic-plastic simulation the coefficients $N(1)$ and $N(2)$ are

$$N(1) = \left\langle \left( 2G - \frac{G^2}{G + 9K\alpha^2 + \frac{1}{\sqrt{3}} I_1\alpha'} \right) \frac{d\epsilon_{12}}{dt} \right\rangle$$
$$= \frac{d\epsilon_{12}}{dt} \left( 2G - \frac{G^2}{G + 9K\alpha^2 + \frac{1}{\sqrt{3}} I_1\alpha'} \right)$$
$$= 0.2365 \text{ MPa/s}$$

$$N(2) = t \left( \frac{d\epsilon_{12}}{dt} \right)^2 Var \left[ 2G - \frac{G^2}{G + 9K\alpha^2 + \frac{1}{\sqrt{3}} I_1\alpha'} \right]$$
$$= 0.0001 t \text{ (MPa/s)}^2$$

It should be noted that for Problem III, since the shear modulus is deterministic, the pre-yield elastic case is deterministic.

### 105.2.8 Probabilistic Elasto-Plasticity: Results and Verifications of Example Problems

In this section results are presented for elastic and elastic–plastic probabilistic 1D problem. The results are obtained by using FPK equation approach described in previous sections and in the companion paper (Jeremić et al., 2007a). In addition to that, the Monte Carlo based verification of developed solutions (results) is presented. The effort to verify developed solutions (that are based on FPK approach) plays a crucial role in presented development of probabilistic elasto–plasticity as there are no previously published solutions which could have been used for verification. In addition to that, verification and validation efforts should always be included in any modeling and simulations work (Oberkampf et al., 2002).

For linear elastic constitutive rate equations (Problem-I and pre-yield case of Problem-II) the verification is performed by comparing solutions obtained through the use of FPK equation approach with high accuracy (exact) solution, using a transformation method of random variables (Montgomery and Runger, 2003). This method is applicable as for rate-independent linear elastic case the 1D shear constitutive equation simplify to a linear algebraic equation of the form,

$$\sigma_{12} = 2G\epsilon_{12} = u(G, \epsilon_{12})$$  \hspace{1cm} (105.39)

Using the definition of strain rate, the above equation can be written in terms of time $t$ as,

$$\sigma_{12} = 2G(0.054t) = v(G, t)$$  \hspace{1cm} (105.40)
where, 0.054 1/s is the arbitrary strain-rate assumed for this example problem. According to the transformation method of random variables (Montgomery and Runger, 2003), and, given the continuous random variable (shear modulus) $G$, with PDF $g(G)$ and Eqs. (105.39) or (105.40) as one-to-one transformations between the values of random variables of $G$ and $\sigma_{12}$, one can obtain the PDF of shear stress ($\sigma_{12}$), $P(\sigma_{12})$ as,

$$P(\sigma_{12}) = g(u^{-1}(\sigma_{12}, \epsilon_{12})) |J|$$

(105.41)

which will allow for predicting the evolution of PDF of $\sigma_{12}$ with $\epsilon_{12}$ or,

$$P(\sigma_{12}) = g(v^{-1}(\sigma_{12}, t)) |J|$$

(105.42)

Eq. (105.42) will predict the evolution of PDF of $\sigma_{12}$ with $t$. In Eqs. (105.41) and (105.42), functions $G = u^{-1}(\sigma_{12}, \epsilon_{12})$ or $G = u^{-1}(\sigma_{12}, t)$ are the inverse of functions $\sigma_{12} = u(G, \epsilon_{12})$ or $\sigma_{12} = v(G, t)$ respectively and $J = du^{-1}(\sigma_{12}, \epsilon_{12})/d\sigma_{12}$ and $J = dv^{-1}(\sigma_{12}, t)/d\sigma_{12}$ are their respective Jacobians of transformations.

For non-linear elastic-plastic constitutive rate equations (post-yield cases of Problems II and III) the verification is done using Monte-Carlo simulation technique by generating sample data for material properties from standard normal distribution and by repeating solution of the deterministic elastic-plastic constitutive rate equation for each data generated above. The probabilistic characteristics of resulting random stress variable for each time (or strain) step are then easily computed. A relatively large number of data points (1,000,000) were generated for each material constant random variable for this simulation purpose.

### 105.2.9 Problem I

The evolution of PDF of shear stress with time and shear strain is shown in Figures 105.3 and 105.4. Presented PDFs are for linear elastic material with random shear modulus, and were obtained using FPE approach (Fig. rfigure:ElasticPDF) and transformation method (Fig. 105.4).

The contours of evolution of PDFs are compared in Fig. 105.5. Similarly, comparison of the evolution of mean and standard deviations are shown in Fig. 105.6. It can be seen from the comparison figure that even though the FPK approach predicted the mean behavior exactly, it slightly over-predicted the standard deviation. This is because of the approximation used to represent the Dirac delta function, which was used as the initial condition for the FPK. One may note that at $\epsilon_{12} = 0$, the probability of shear stress $\sigma_{12}$ should theoretically be 1 i.e. all the probability mass should theoretically be concentrated at $\sigma_{12} = 0$. As such, it would be best described by the Dirac delta function. However, for numerical simulation of FPK, Dirac delta function as initial condition was approximated with a Gaussian function of
mean zero and standard deviation of $0.00001$ MPa, as shown in Fig. ??

This error in the initial condition advected and diffused into the domain with the simulation of the evolution process. This error could be minimized by better approximating the Dirac delta initial condition (but at higher computational cost). The effect of approximating the initial condition of the PDF of shear stress at $\epsilon_{12} = 0.0426\%$ is shown in Fig. 105.7. In this figure the actual PDF at $\epsilon_{12} = 0.0426\%$ obtained using the transformation method was compared with the PDFs at $\epsilon_{12} = 0.0426\%$ obtained using the FPK approach with three different approximate initial conditions - all having zero mean but standard deviations of $0.01$ MPa, $0.005$ MPa and $0.00001$ MPa.

One may also note that finer approximation of initial condition necessitates finer discretization of stress domain close to (or at) $\sigma_{12} = 0$. The finite difference discretization scheme adopted here uses the same fine discretization uniformly all throughout the entire domain. It is noted that that fine, uniform discretization is not needed (and is quite expensive) in later stages of calculation of evolution of PDF, but is kept the same for simplicity sake. In presented examples, to properly capture the approximate initial condition (as shown in Fig. ??), the stress domain between $-0.1$ MPa and $+0.1$ MPa was discretized with a uniform step size of $0.000005$ MPa and hence there is a total of $40,000$ nodes. This not only requires
large computational effort but is also very memory sensitive. An adaptive discretization technique will be a much better approach to solving this problem. Current work is going on in formulating an adaptive algorithm for the solution of this type of problem.

105.2.10 Problem II

The solution to this problem involves the solving two FPK equations, one corresponding to the pre-yield elastic part and the other corresponding to the post-yield elastic-plastic part. The elastic part of this problem is identical to Problem I. The initial condition for the post-yield elastic-plastic part of the problem is random and is shown in Fig. 105.8. It may be noted that this initial condition corresponds to the PDF of shear stress \( P(\sigma_{12}) \) at yield obtained from the solution of FPK equation of the pre-yield elastic part. A view of the surface of evolution of the PDF of shear stress versus shear strain (time) is shown in Fig. 105.9. Another view to the PDF of stress–strain surface is shown in Fig. 105.10. It is noted that the yielding of this material occurred at \( t=0.00789 \) second (which is equivalent to \( \epsilon_{12}=\]
Figure 105.5: Comparison of Contours of Time (or Strain) Evolution of Probability Density Function for Shear Stress for Elastic Constitutive Rate Equation with Random Shear Modulus (Problem–I) for FPE Solution and Variable Transformation Method Solution.

0.0426 %). The evolution contours for PDF of shear stress versus strain (time) along with the mean and standard deviations are shown in Fig. 105.11. It can be seen from that figure that, as expected, the evolution of mean of shear stress changes slope after the material yielded. Another interesting aspect to note is the relative slope of the evolution of standard deviation with respect to the evolution of mean. The relative slope in the pre-yield elastic zone increases at a higher rate during the evolution process when compared with that in the post-yield elastic-plastic zone. In other words, in the evolution process the post-yield elastic–plastic constitutive rate equation did not amplify the initial uncertainty as much as the pre-yield elastic constitutive rate equation did. This can be easily viewed from Fig. 105.12 where the post-yield elastic-plastic evolution of PDF of shear stress was compared with fictitious extension of elastic evolution of PDF. Comparing the PDF of shear stress at $\epsilon_{12} = 0.0804\%$ (which is equivalent to $t = 0.01489s$), one can conclude that the variance of predicted elastic-plastic shear stress is much smaller (i.e. prediction is less uncertain) as compared to the same if the material were modeled as completely elastic.

Fig. 105.13 compares the evolution of means and standard deviations of predicted shear stress obtained using FPK equation approach and transformation method (pre-yield behavior)/Monte-Carlo approach (post-yield behavior). Although in the pre–yield response the FPK equation approach over-predicted the evolution of standard deviations because of reasons discussed earlier, in the post-yield
response it matched closely at regions further from the yielding region. The somewhat larger difference between FPK equation solution and the verification one (Monte Carlo solution) close to the yielding region is attributed to the fact that the initial condition for solution of post-yield elastic-plastic FPK equation was obtained from the solution of pre-yield elastic FPK equation. One way to better predict the overall probabilistic elastic-plastic behavior, would probably be to obtain the pre-yield elastic behavior through the transformation method and then use the FPK approach to predict post-yield elastic-plastic behavior.

105.2.11 Problem III

In this problem, the pre–yield linear elastic part is deterministic, however, at yield there is a distribution (with very small standard deviation) in shear stress due to assumed distribution in yield parameter $\alpha$. The distribution in shear stress corresponds to the PDF of the random variable $\alpha I_1$ (first invariant of the stress tensor or mean confining stress) and is assumed to be deterministic. This PDF of shear stress at yield was assumed to be the initial condition for the solution of post-yield elastic-plastic FPK equation.
The evolution of PDF for shear stress versus strain (time) is shown in Fig. 105.15. In addition to that the contours (including mean and standard deviation) of the evolution of PDF for shear stress versus strain (time) are shown in Fig. 105.16.

Looking at Fig. 105.16 and comparing the slopes of evolution of mean and standard deviation, one can conclude that the elastic-plastic evolution process didn’t amplify the initial uncertainty in yield strength significantly. The initial (at yield) probability density function of shear stress just advected into the domain during the elastic–plastic evolution process without diffusing much. Fig. 105.15 clearly shows this advection process. The evolution of mean and standard deviations of shear stress obtained from the FPK equation approach was compared with those obtained from the Monte Carlo simulation and is shown in Fig. 105.17.
105.3 Probabilistic Yielding and Cyclic Loading

Modeling of geomaterials is inherently uncertain. These uncertainties stem from natural variability of geomaterials (spatial uncertainty), and testing and transformation errors (point uncertainty) (Lacasse and Nadim (1996), Phoon and Kulhawy (1999a)). These uncertainties not only affect the failure characteristics of geomaterials, but also the behavior of geostructures, made with geomaterials. Traditionally, geotechnical engineering community deals with uncertainties in geomaterial by applying (large) factor of safety. However, use of large factors of safety results not only in over-expensive design, but also, sometimes, in unsafe structures (cf. Duncan (2000b)). Hence, in recent years, the geotechnical community has seen an increasing emphasis on probabilistic characterization of soil and subsequent reliability-based design.

One of the important aspects of probabilistic geomechanics simulation that has received less attention is the probabilistic constitutive problem. Among the few published papers were those by Fenton and Griffiths (Fenton and Griffiths (2002), Fenton and Griffiths (2003), Fenton and Griffiths (2005)) on
Figure 105.9: Evolution of PDF of shear stress versus strain (time) for elastic-plastic material with random shear modulus (Problem–II). View 1.

probabilistic simulation of spatially random c-ϕ soil using Monte Carlo technique, and those by Anders and Hori (Anders and Hori (1999), Anders and Hori (2000)) on probabilistic simulation of von Mises elastic-perfectly plastic material using perturbation technique. Both Monte Carlo and perturbation techniques have their inherent drawbacks (Matthies et al. Matthies et al. (1997), Keese Keese (2003)) and in dealing with those, recently, Jeremić et al. Jeremić et al. (2007b) proposed Eulerian–Lagrangian form of Fokker–Planck–Kolmogorov equation (FPKE) approach (cf. Kavvas Kavvas (2003)) to modeling and simulation for probabilistic elasto–plasticity. FPKE approach to probabilistic elasto–plasticity not only overcomes the drawbacks associated with other probabilistic simulation techniques, but also is fully compatible with the incremental theory of elasto–plasticity, and hence can easily be applied to probabilistic modeling and simulation of different elastic–plastic constitutive models. Solution strategies for FPK partial differential equation, corresponding to elastic–plastic constitutive rate equation and simulated probabilistic stress-strain responses under monotonic loading, assuming mean stress yielding, were discussed by Sett et al. (Sett et al. (2007c), Sett et al. (2007d)) for both linear and non-linear
hardening models. The concept of probabilistic yielding was introduced and its effect on constitutive simulation under monotonic loading was discussed by Jeremić and Sett (2009b). It was shown that due to uncertainty in yield function (stress), there is always a possibility, depending upon the magnitude of uncertainty, that plastic behavior starts at very very low strain and influence of elastic behavior continues far into plastic domain (at large strains) and hence, the ensemble average (mean) of all the possibilities or the most probable (mode) possibility differ from deterministic behavior. In addition to that, a very realistic, smooth transition between elastic and plastic domains was observed even for elastic perfectly plastic models. Further, nonlinear behavior was observed even for linear hardening models.

In this paper, the concept of probabilistic yielding is extended to 1-D cyclic simulations of geomaterials. Both elastic–perfectly plastic and hardening-type material model are considered. The numerical technique of solving FPKE cyclically with probabilistic yielding is discussed. Simulated responses were discussed in terms of probability density function (PDF) and its statistical moments.

Modeling of geomaterials is inherently uncertain. This uncertainty stems from natural variability of geomaterials (spatial uncertainty), and testing and transformation errors (point uncertainty) (Lacasse...
Figure 105.11: Contour of evolution of PDF for shear stress versus strain (time) for elastic-plastic material with random shear modulus (Problem–II).

and Nadim Lacasse and Nadim (1996), Phoon and Kulhawy Phoon and Kulhawy (1999a)). These uncertainties not only affect the failure characteristics of geomaterials, but also the behavior of geostuctures, made with geomaterials. Traditionally, geotechnical engineering community deals with uncertainties in geomaterial by applying (large) factor of safety. However, use of large factors of safety results not only in over-expensive design, but also, sometimes, in unsafe structures (cf. Duncan Duncan (2000b)). Hence, in recent years, the geotechnical community has seen an increasing emphasis on probabilistic characterization of soil and subsequent reliability-based design.
Figure 105.12: Comparison of evolution of PDF for elastic-plastic material and extended elastic material cases for random shear modulus.
One of the important aspects of probabilistic geomechanics simulation that has received less attention is the probabilistic constitutive problem. Among the few published papers were those by Fenton and Griffiths (Fenton and Griffiths (2002), Fenton and Griffiths (2003), Fenton and Griffiths (2005)) on probabilistic simulation of spatially random $c$-$\phi$ soil using Monte Carlo technique, and those by Anders and Hori (Anders and Hori (1999), Anders and Hori (2000)) on probabilistic simulation of von Mises elastic-perfectly plastic material using perturbation technique. Both Monte Carlo and perturbation techniques have their inherent drawbacks (Matthies et al. Matthies et al. (1997), Keese Keese (2003)) and in dealing with those, recently, Jeremić et al. Jeremić et al. (2007b) proposed Eulerian–Lagrangian form of Fokker–Planck–Kolmogorov equation (FPKE) approach (cf. Kavvas Kavvas (2003)) to modeling and simulation for probabilistic elasto–plasticity. FPKE approach to probabilistic elasto–plasticity not only overcomes the drawbacks associated with other probabilistic simulation techniques, but also is fully compatible with the incremental theory of elasto–plasticity, and hence can easily be applied to probabilistic modeling and simulation of different elastic–plastic constitutive models. Solution strategies for FPK partial differential equation, corresponding to elastic–plastic constitutive rate equation and simulated probabilistic stress-strain responses under monotonic loading, assuming mean stress yielding, were discussed by Sett et al. (Sett et al. (2007c), Sett et al. (2007d)) for both linear and non-linear
hardening models. The concept of probabilistic yielding was introduced and its effect on constitutive simulation under monotonic loading was discussed by Jeremić and Sett Jeremić and Sett (2009b). It was shown that due to uncertainty in yield function (stress), there is always a possibility, depending upon the magnitude of uncertainty, that plastic behavior starts at very very low strain and influence of elastic behavior continues far into plastic domain (at large strains) and hence, the ensemble average (mean) of all the possibilities or the most probable (mode) possibility differ from deterministic behavior. In addition to that, a very realistic, smooth transition between elastic and plastic domains was observed even for elastic perfectly plastic models. Further, nonlinear behavior was observed even for linear hardening models.

In this paper, the concept of probabilistic yielding is extended to 1–D cyclic simulations of geomaterials. Both elastic–perfectly plastic and hardening-type material model are considered. The numerical technique of solving FPKE cyclically with probabilistic yielding is discussed. Simulated responses were discussed in terms of probability density function (PDF) and its statistical moments.

Figure 105.14: Initial condition for FPK equation for elastic–plastic material with random yield strength (Problem–III).
105.3.1 Fokker–Planck–Kolmogorov Approach to Probabilistic Elasto–Plasticity

The Eulerian–Lagrangian form Fokker–Planck–Kolmogorov equation (cf. Kavvas Kavvas (2003)) corresponding to generalized 1–D constitutive rate equation can be written as (Jeremić et al. Jeremić et al. (2007b)):

\[
\frac{\partial P(\sigma(x,t),t)}{\partial t} = \frac{\partial}{\partial \sigma} \left\{ \left[ \eta(\sigma,D,e;x,t) \right] + \int_0^t d\tau Cov_0 \left[ \frac{\partial \eta(\sigma,D,e;x,t)}{\partial \sigma} ; \eta(\sigma,D,e;x_{t-\tau},t-\tau) \right] \right\} P(\sigma(x,t),t) \\
+ \frac{\partial^2}{\partial \sigma^2} \left\{ \int_0^t d\tau Cov_0 \left[ \eta(\sigma,D,e;x,t) ; \eta(\sigma,D,e;x_{t-\tau},t-\tau) \right] \right\} P(\sigma(x,t),t)
\]

where, \( P(\sigma(x,t),t) \) is the probability density of stress \( \sigma \) at (pseudo) time \( t \), and \( \eta \) is the operator variable, obtained by collecting together all the operators and variables on the r.h.s of the generalized
Figure 105.16: Contour PDF for shear stress versus strain (time) for elastic–plastic material with random yield strength (Problem–III).

constitutive rate equation:

\[ \frac{d\sigma(x_t, t)}{dt} = \eta(\sigma, D, \epsilon; x_t, t) \]  \hspace{1cm} (105.44)

In Eq. (105.44), \( \epsilon \) is the strain, and \( D \) is the tangent modulus, which could be elastic or elastic–plastic:
Figure 105.17: Comparison of Mean and Standard Deviation of Shear Stress for Elastic-Plastic Constitutive Rate Equation with Random Yield Strength (Problem-III) for FPE Solution and Monte Carlo Simulation Solution

\[
D = \begin{cases} 
D^{\text{el}} & \text{elastic} \\
D^{\text{el}} - \frac{D^{\text{el}} \partial U \partial f}{\partial \sigma \partial \sigma} D^{\text{el}} & \text{elastic-plastic} 
\end{cases}
\]

where, \( D^{\text{el}} \), \( f \), \( U \), \( q_* \), and \( r_* \) are elastic modulus, yield surface, plastic potential surface, internal variable(s), and rate(s) of evolution of internal variable(s) respectively.

Eq. (105.43) is the most general form of elastic–plastic constitutive rate equation, written in probability density space. This equation (Eq. (105.43)) can be written in a more compact form:

\[
\frac{\partial P(\sigma(x,t),t)}{\partial t} = \frac{\partial}{\partial \sigma} \left\{ N_{(1)} P(\sigma(x,t),t) \right\} + \frac{\partial^2}{\partial \sigma^2} \left\{ N_{(2)} P(\sigma(x,t),t) \right\}
\]

where, \( N_{(1)} \) and \( N_{(2)} \) are advection and diffusion coefficients respectively, and are material model specific. By specializing Eq. (105.46) to (any) particular constitutive model, the resulting FPKE can be
solved to obtain the probability density function of stress response, given uncertainties in material properties and driving strain. However, difference in material behavior in elastic and elastic-plastic regions necessitates solution of FPKE twice - one corresponding to elastic constitutive equation (with $N_{el}^{(1)}$ and $N_{el}^{(2)}$, the advection and diffusion coefficients corresponding to elastic constitutive equation) and the other corresponding to elastic–plastic constitutive equation (with $N_{ep}^{(1)}$ and $N_{ep}^{(2)}$, the advection and diffusion coefficients corresponding to elastic–plastic constitutive equation). The switch from elastic to elastic–plastic region (solution) can be controlled using mean stress yielding:

\[
\text{if } \langle f \rangle < 0 \lor (\langle f \rangle = 0 \land d \langle f \rangle < 0) \quad \text{use elastic FPKE}
\]

\[
\text{or, if } \langle f \rangle = 0 \lor d \langle f \rangle = 0 \quad \text{use elastic–plastic FPKE}
\]

However, difficulty arises if the material yield parameter(s) are uncertain, as the mean yield criteria then does not account for the complete probabilistic yielding of material. For example, such mean yielding will neglect the possibilities of elastic–plastic behavior in the elastic region and vice versa. The concept of probabilistic yielding overcomes this limitation, as it solves Eq. (105.46) once, with equivalent advection and diffusion coefficients, $N_{eq}^{(1)}$ and $N_{eq}^{(2)}$ (Jeremič and Sett (2009b)):

\[
N_{eq}^{(1)}(\sigma) = (1 - P[\Sigma_y \leq \sigma])N_{el}^{(1)} + P[\Sigma_y \leq \sigma]N_{ep}^{(1)}
\]

\[
N_{eq}^{(2)}(\sigma) = (1 - P[\Sigma_y \leq \sigma])N_{el}^{(2)} + P[\Sigma_y \leq \sigma]N_{ep}^{(2)}
\]

(105.48)

where $(1 - P[\Sigma_y \leq \sigma])$ represents the probability of material being elastic, while $P[\Sigma_y \leq \sigma]$ represents the probability of material being elastic–plastic. The probabilities of material being elastic and the probabilities of material being elastic–plastic can easily be calculated from the cumulative density function of yield function (stress).

It is worth noting that the probabilistic yield criterion (Eq. (105.48)) represents probabilistic restatement of the deterministic yield criteria. The probabilistic yield criteria is introduced (or, the deterministic yield criteria is written in probability space) in order to properly model uncertain (probabilistic) yield strength.

It is also very interesting to note that proposed approach for calculating equivalent advection and diffusion coefficients is similar to the solution strategy of famous Black–Scholes Black and Scholes (1973) equation in financial engineering modeling of European option, where probabilities of exercise of the (European) option, obtained from cumulative density functions, are multiplied with stock price and present value of option strike price to calculate the option price.
105.3.2 Elastic–Perfectly Plastic Material

In this section, the FPKE–approach, along with the concept of probabilistic yielding, is applied to simulate 1–D (shear stress–shear strain) cyclic behavior of elastic–perfectly plastic material. Only von Mises material model has been considered. It may, however, be noted that presented development is general enough to be used with any material model and that von Mises is just one such model we use for illustration purposes.

The von Mises yield criteria can be written as:

\[ \sqrt{J_2} - k = 0 \]  

(105.49)

where, \( k \) is a material parameter (yield strength like) and \( J_2 = 3/2 s_{ij}s_{ij} \) is the second invariant of deviatoric stress tensor \( s_{ij} = \sigma_{ij} - 1/3 \sigma_{kk} \delta_{ij} \). For 1–D shear, Eq. (105.49) becomes:

\[ |\sigma| - \sigma_y = 0 \quad \text{or} \quad \sigma = \pm \sigma_y \]  

(105.50)

The yielding occurs at a yield stress of \( \pm \sigma_y \). It, however, is important to note that both \( \sigma_y \) and \( \sigma \) are uncertain and are described by their respective probability density functions. For elastic–perfectly plastic material, the distribution of yield stress (\( \sigma_y \)) is given by its experimentally measured initial distribution, and remains constant. The stress (\( \sigma \)), however, evolves according to the governing FPKE (Eq. (105.46)) and its distribution is given by the solution of the governing FPKE (Eq. (105.46)). For 1–D von Mises elastic–perfectly plastic shear constitutive model, the elastic and the elastic–plastic advection and diffusion coefficients of the governing FPKE (Eq. (105.46)), becomes:

\[ N_{el}^{(1)} = \frac{de_{xy}}{dt} \langle G \rangle ; \quad N_{el}^{(2)} = t \left( \frac{de_{xy}}{dt} \right)^2 \text{Var}[G] \]
\[ N_{ep}^{(1)} = 0 \quad ; \quad N_{ep}^{(1)} = 0 \]  

(105.51)

where, \( G \) is the shear modulus, \( de_{xy} \) is the (deterministic) incremental shear strain, \( t \) is the pseudo time, \( \langle \cdot \rangle \) represents expectation operation and \( \text{Var}[\cdot] \) represents variance operation. The equivalent advection and diffusion coefficients (refer Eq. (105.48)) for von Mises elastic–perfectly plastic material, then, becomes:

\[ N_{eq}^{(1)}(\sigma) = (1 - P[\Sigma_y \leq \sigma]) \frac{de_{xy}}{dt} \langle G \rangle \]
\[ N_{eq}^{(2)}(\sigma) = (1 - P[\Sigma_y \leq \sigma]) t \left( \frac{de_{xy}}{dt} \right)^2 \text{Var}[G] \]  

(105.52)

One may note that, in deriving the elastic and elastic–plastic advection and diffusion coefficients (Eq. (105.51)), it was assumed that spatial random field material properties (\( G \), and \( \sigma_y \)) would be
first discretized into random variables, for example at Gauss points, by appropriate tools, for example Karhunen–Loève expansion (Karhunen Karhunen (1947), Loève Loève (1948), Ghanem and Spanos Ghanem and Spanos (1991)). In other words, the solution of FPKE, with advection and diffusion coefficients given by Eq. (105.52), represents point–location scale von Mises elastic–perfectly plastic material behavior, and not the local–average material behavior. The local–average material behavior, if sought for, can then be assembled using polynomial chaos expansion (Wiener Wiener (1938), Ghanem and Spanos Ghanem and Spanos (1991)).

105.3.2.1 Probability Density Function

The FPKE (Eq. (105.46)), with advection and diffusion coefficients given by Eq. (105.52), was solved incrementally with pseudo time steps using method of lines. The stress domain of the Fokker–Planck–Kolmogorov PDE was discretized first on a uniform grid by central differences, and thereby obtaining a series of ODE. The series of ODEs was then solved, after incorporating boundary conditions, simultaneously and incrementally, with $n$ pseudo time steps, using a standard open–source ODE solver, SUNDIALS Hindmarsh et al. (2005), which utilizes ADAMS method and functional iteration.

The yield shear strength ($\sigma_y$) of the material was assumed to have a mean value of 60 kPa with a COV of 30%, values typical for clay (Federal Highway Administration Federal Highway Administration (2002), Lacasse and Nadim Lacasse and Nadim (1996)). Also, the yield shear strength was assumed to be either normal or Weibull (with shape parameter of 3.31 and scale parameter of 0.067) distribution as shown in Fig. 105.18. The shear modulus ($G$) was also assumed to be either normal or Weibull distribution, but

![Figure 105.18: Elastic–perfectly plastic probabilistic model: PDF of yield stress](image-url)
with a mean value of 100 MPa and a COV of 25%. The cyclic probabilistic von Mises, elastic–perfectly plastic shear stress–shear strain response (evolutionary probability density function (PDF) of shear stress), for the case where both yield shear strength ($\sigma_y$) and shear modulus ($G$) are normally distributed, is shown in Fig. 105.19. Two different views of the loading–unloading–reloading cycle are shown, focusing on the transition between loading and unloading, and unloading and reloading branches. As can be seen from Fig. 105.19, PDF for initial stress (a deterministic Dirac delta function at stress–strain origin) advected and diffused into the domain, governed by the advection and diffusion coefficients (Eq. (105.52)). It is very important to also note that, even–though the deterministic response for von Mises elastic–perfectly

![Figure 105.19](image_url)

**Figure 105.19**: Elastic–perfectly plastic probabilistic model under cyclic loading: evolutionary PDF of shear stress (a) view from the junction of loading and unloading branches (probability densities of shear stress are truncated at a value 1500 for clarity of the plot) and (b) view from the junction of unloading and reloading branches (probability densities of shear stress are truncated at a value of 150 for clarity of the plot)
plastic material is bi-linear, due to introduced uncertainties in yielding, the probabilistic response is non-linear from the beginning. That is, due to uncertainty in yield strength, there is a (small) possibility that the material becomes elasto-plastic from the very beginning of loading. This possibility has been quantified from the PDF of the yield strength and taken into consideration implicitly during simulation using the equivalent advection and diffusion coefficients (\( N_{eq}^{(1)} \) and \( N_{eq}^{(2)} \), refer Eq. (105.52)). Those coefficients assigns probability weights to the realizations of stress response based on the probability of material being elastic or elastic-plastic. Initially, in the loading branch, at small strains, the probability of material being elastic-plastic is very small and hence, the initial probabilistic stress response (ensemble of all realizations) is closer (but not fully) to linear, elastic response. However, as strain increases, the probability of elastic-plastic material behaving increases and the probabilistic stress response gradually becomes more elastic-plastic (Fig. 105.19(a)).

Upon unloading, the material behaves as (mostly) elastic since elastic-plastic probability weights from the governing PDF of mirror image (negative) of shear strength (Fig. 105.18) are initially very small. During later stages of unloading (loading in the opposite direction), and similar to the loading branch, the elastic-plastic probability weights increase and gradually transition the response toward elasto-plasticity (Fig. 105.19(b)). Similar to this, in the subsequent reloading branch, the probability weights are again governed the PDF of (positive, loading branch of) shear strength (Fig. 105.18), and hence the probabilistic response is again initially more linear, elastic, while gradually it transitions to full elasto-plasticity.

105.3.2.2 Case of Increasing Strain Loops

In Fig. 105.20, the evolutionary PDF of shear stress for von Mises elastic-perfectly plastic material (refer Fig. 105.19) is plotted in terms of its statistical moments – the evolutionary mean (Fig. 105.20(a)), and standard deviation (Fig. 105.20(b)) of shear stress – for the first couple of cycles with increasing strain loops. The mean response, when both the yield shear strength (\( \sigma_y \)) and the shear modulus (\( G \)) are modeled as Weibull distribution, is also shown in Fig. 105.20(a). The oscillations in the evolution of standard deviation of shear stress with shear strain are due to step size issue, inherent to the forward Euler method that has been used in solving the FPKE. Work is underway to implement linearly implicit mid-point rule for solving the FPKE corresponding to elastic-plastic constitutive rate equation.

The very important observation that can be made using Fig. 105.20(a) is that, if one consider uncertainties in geomaterial properties, even the simplest elastic-perfectly model, captures some of the very important features of geomaterial behaviors. For example, reduction of (secant) modulus with cyclic strain, commonly observed in soil (cf. Vucetic and Dobry Vucetic and Dobry (1991)), is fairly nicely captured. If using deterministic models, this feature can only be somewhat successfully modeled with
fairly complex models, which require many more parameters. It is important to remark that for our probabilistic modeling, (only) statistical distributions (probability density functions) of shear modulus ($G$) and shear strength ($\sigma_y$), are needed. Expansion of elastic–plastic modeling into probability space seems to have added significant new capabilities to modeling.

105.3.2.3 Case of Constant Strain Loops

This von Mises elastic–plastic material, however, didn’t exhibit (secant) modulus degradation, commonly observed in clay (cf. Vucetic and Dobry (1988)), when the material is cycled repeatedly at the same strain. Fig. 105.21(a) shows such probabilistic response (mean of shear stress). The material was cycled repeatedly up to 0.2% strain. Only first three cycles are shown. It is important to note that the von Mises mean elastic–plastic material behavior is function of both the mean and standard deviation of both shear modulus ($G$) and yield shear strength ($\sigma_y$). The same von Mises elastic–perfectly plastic model with a different set of material properties could, however, be able to capture the degradation of mean (secant) shear modulus. For example, Japanese stiff clay, when modeled as von Mises elastic–perfectly plastic material, exhibited modulus degradation with number of cycles (Sett et al. (2008)).

105.3.2.4 Monotonic Loading

For completeness of comparison, the monotonic behavior of this probabilistic von Mises perfectly plastic material is also shown (refer Fig. 105.22). As can be observed from Fig. 105.22(a), the mean shear
Figure 105.21: Elastic–perfectly plastic probabilistic model under cyclic loading with all equal loops: evolution of (a) mean and (b) standard deviation of shear stress

stress non–linearly increases with shear strain before reaching the perfectly plastic state.

Figure 105.22: Elastic–perfectly plastic probabilistic model under monotonic loading: evolution of (a) mean, (b) standard deviation, and (c) mean ± standard deviation of shear stress

Physically, one may visualize the probabilistic soil constitutive response as an ensemble of the behaviors of infinite number of soil particles in a representative volume element (RVE), for example, a laboratory soil specimen. The behavior of an individual soil particle in a RVE is governed, in case of elastic–perfectly plastic material, by its modulus and strength. However, if the modulus and strength of each particle are different, for example, governed by their respective PDF, then each particle would behave differently. The PDF of the response behavior then represents the ensemble of all such behaviors, with their respective probability weights. The mean, on the other hand, represents the ensemble average of all such behaviors. In this context, it is important to note that the behaviors presented in this paper do not take into account the correlation between soil particles (scale effect). The scale effect
can be accounted for, among others, using stochastic elastic–plastic finite element technique. Sett Sett (2007) proposed one such finite element method by extending the spectral approach to stochastic finite element (cf. Ghanem and Spanos Ghanem and Spanos (1991)) to elastic–plastic problems by updating the material properties at Gauss integration points using the FPKE approach, as the material plastifies.

Further to the promise of an alternate approach to geomaterial modeling, probabilistic approach also quantifies our confidence in the simulated behavior of geomaterials. FPKE based probabilistic elastoplasticity solves for second-order accurate evolutionary PDF of shear stress (Fig. 105.19). Ability to obtain the PDF of stress accurately is very important in failure simulation of geomaterials, as they often fail at low probabilities (tails of PDF). A full PDF contains enormous amount of information. From the PDF, other than the statistical moments, other useful engineering information, for example, the probability of exceedance, most probable solution, as well as some derivative application like sensitivity analysis can be easily obtained or derived. Figs. 105.20(b) and 105.21(b) show one of the important confidence measuring parameters, the evolutionary standard deviation of shear stress (square-root of second moment of the evolutionary PDF of shear stress (Fig. 105.19)), for cyclic responses with increasing loops and all equal loops, respectively. As can be observed from the above figures (Figs. 105.20(b) and 105.21(b)), inside any branch (loading, unloading, re–loading, re–unloading, ...), as well as in Fig. 105.22(b), where the monotonic response is shown, the standard deviation, first increases and then decreases. This is because, initially, when the material is mostly elastic, both the uncertainties in shear modulus ($G$) and yield strength ($\sigma_y$) are governing. As material becomes mostly elastic–plastic, the influence of uncertainty in shear modulus ($G$) decreases. However, it is important to note that this type of standard deviation response is not generic to all von Mises elastic–perfectly plastic material. The standard deviation response is very much dependent on the amount uncertainties present in both shear modulus ($G$) and yield strength ($\sigma_y$). For example, Fig. 105.23(b), shows probabilistic response of cyclic behavior of the same material model, except that COV of yield strength ($\sigma_y$), is now assumed to be 300%. The standard deviation response shown here is always increasing which is completely different from what was observed in previous case (Figs. 105.20(b), 105.21(b) and 105.22(b))). This is because, for this material, the COV of shear modulus (assumed 30%) is non–significant, compared to the COV of yield strength (assumed 300%), and hence, the standard deviation response (Fig. 105.23(b)) is predominantly influenced by the uncertainty in yield strength ($\sigma_y$). Similar standard deviation response can be observed in Fig. 105.24(b), where the material with large COV of yield strength was subjected to monotonic loading.

It is also interesting to compare Figs. 105.21(a) and 105.23(a). Both are mean responses of von Mises elastic–perfectly plastic material model with same material parameters, except with different COV of yield strength. COV of yield strength for simulation in Fig. 105.21(a) was 30% and that for simulation in Fig. 105.23(a) was 300%. It is observed that a completely different responses were obtained. The effect
Figure 105.23: Elastic–perfectly plastic probabilistic model under cyclic loading with all equal loops (probabilistic model parameters are exactly the same as used for simulation in Fig. 105.21, but with very large yield uncertainty): evolution of (a) mean and (b) standard deviation of shear stress

Figure 105.24: Elastic–perfectly plastic probabilistic model under monotonic loading (model parameters are exactly the same as used for simulation in Fig. 105.22, but with very large yield uncertainty): evolution of (a) mean, mode, (b) standard deviation, and (c) mean ± standard deviation of shear stress

of COV of yield strength on monotonic mean behavior can, similarly, be compared in Figs. 105.22(a) and 105.24(a).

105.3.2.5 Hardening Material

In this section, the influence of probabilistic yielding is evaluated on cyclic responses of isotropic and kinematic hardening materials. To this end, the same example, as discussed in the previous section (Section 105.3.2) is used but with appropriate hardening rule – isotropic or kinematic.

The main difference between the simulations shown in Section 105.3.2 for elastic–perfectly plastic...
material is that for a hardening material the internal variables ($q^*$, refer Eq. (105.45)) will evolve as the material plastifies. Such evolution (change) of internal variables is here assumed to be a function of plastic strain. The FPKE that govern the probabilistic evolution of internal variable ($q$) can be written, in most the general form, as:

$$
\frac{\partial P(q(x_t,t),t)}{\partial t} = \frac{\partial}{\partial q} \left\{ N_{(1)}^{eq}(q(x_t,t),t) \right\} + \frac{\partial^2}{\partial q^2} \left\{ N_{(2)}^{eq}(q(x_t,t),t) \right\} \tag{105.53}
$$

where, $N_{(1)}^{eq}$ and $N_{(2)}^{eq}$ are the equivalent advection and diffusion coefficients, respectively, for the internal variable. As explained for the case of probabilistic stress response for elastic–perfectly plastic material (refer Section 105.3.2), since point–location scale FPKE will be solved, the equivalent advection and diffusion coefficients for the internal variable, $N_{(1)}^{eq}$ and $N_{(2)}^{eq}$, can be written as:

$$
N_{(1)}^{eq}(q) = P[\Sigma_y \leq \sigma(q)] \left( \frac{d\epsilon_{xy}}{dt} \right) \left( G + \frac{1}{\sqrt{3}}r \right) \tag{105.54}
$$

$$
N_{(2)}^{eq}(q) = P[\Sigma_y \leq \sigma(q)] t \left( \frac{d\epsilon_{xy}}{dt} \right)^2 Var \left( G + \frac{1}{\sqrt{3}}r \right) \tag{105.55}
$$

where, $r$ is the rate of evolution of internal variable ($q$) with plastic strain. One may note that in the above equivalent advection and diffusion coefficients (Eq. (105.54)), the contributions of probability weights that the material being elastic are absent. This is because the evolution rule of internal variable is governed by the plastic component of strain only. The equivalent advection and diffusion coefficients for shear stress ($N_{(1)}^{eq}$ and $N_{(2)}^{eq}$) for hardening–type materials, will have contributions from both elastic and plastic components, just like the elastic–perfectly plastic case. However, unlike the elastic–perfectly plastic case, those ($N_{(1)}^{eq}$ and $N_{(2)}^{eq}$) will contain the hardening terms:

$$
N_{(1)}^{eq}(\sigma) = \frac{d\epsilon_{xy}}{dt} \left( (1 - P[\Sigma_y \leq \sigma]) \langle G \rangle + P[\Sigma_y \leq \sigma] \left( G - \frac{G^2}{G + \frac{1}{\sqrt{3}}r} \right) \right) \tag{105.56}
$$

$$
N_{(2)}^{eq}(\sigma) = t \left( \frac{d\epsilon_{xy}}{dt} \right)^2 \left( (1 - P[\Sigma_y \leq \sigma]) Var[G] + P[\Sigma_y \leq \sigma] Var \left( G - \frac{G^2}{G + \frac{1}{\sqrt{3}}r} \right) \right) \tag{105.57}
$$

To obtain the probabilistic response of von Mises hardening material, the FPKE for probabilistic evolution of internal variable (Eq. (105.53), with advection and diffusion coefficients given by Eq. (105.54)) needs to be solved incrementally. This solution needs to be done simultaneously with the FPKE for probabilistic evolution of shear stress (Eq. (105.46), with advection and diffusion coefficients given by Eq. (105.55)). Those, in turn, need also to be solved incrementally, with the yield strength random variable ($\Sigma_y$) in Eqs. (105.54) and (105.55) being updated after each incremental step.
105.3.2.6 Isotropic Hardening

For von Mises isotropic hardening material, the yield strength ($\sigma_y$) is the internal variable. Yield strength will evolve probabilistically with plastic strain, following Eq. (105.53), with advection and diffusion coefficients given by Eq. (105.54). The shear stress, on the other hand, evolves in accordance with Eq. (105.46), with advection and diffusion coefficients given by Eq. (105.55).

Fig. 105.25 shows the evolutionary mean and standard deviation of shear stress during first couple of loading–unloading cycles for von Mises isotropic hardening material with a non–dimensional rate of evolution of internal variable (yield strength, in this case) of 10. All other material parameters are assumed to be the same as used for simulation of elastic–perfectly plastic material in the previous section (Section 105.3.2).

![Diagram](a) ![Diagram](b)

Figure 105.25: Isotropic hardening probabilistic model under cyclic loading with increasing loops: evolution of (a) mean and (b) standard deviation of shear stress

The evolved PDFs of yield strength after each branch (loading, unloading, re-loading, and re-unloading) are shown in Fig. 105.26. The initial PDFs of yield strength (positive for loading branch and negative for unloading branch) are the same as assumed for elastic–perfectly plastic material in Section 105.3.2 (refer Fig. 105.18). As expected (and prescribed by the isotropic hardening model), the yield strength evolved (grew) isotropically. However, it is interesting to note the change in probability distributions of yield strength. The normally distributed initial PDFs of yield strength (Fig. 105.18) evolved into much dispersed non-Gaussian distributions having low kurtosis. In other words, when the material is cycled through loading–unloading cycles, the uncertainty in yield strength increases.
Figure 105.26: Isotropic hardening probabilistic model under cyclic loading with increasing loops: evolved PDF of yield stress after (a) loading branch, (b) unloading branch, (c) re-loading branch, and (d) re-unloading branch.

Mathematically, the increase in uncertainty of shear strength is due to the nonlinearity in formulation of probabilistic yielding, that is, the state variable $q$ appears in both advection and diffusion equations (refer Eq. (105.54)), and in the evolution equation for internal variable (Eq. (105.53)).

When comparison is made between Figs. 105.25 and 105.20, one can clearly see that, in simulating cyclic behaviors of geomaterials, isotropic hardening model (Fig. 105.25) performed, as expected, poorly. That is, the elastic–perfectly plastic probabilistic model (Fig. 105.20) captures (PDF of) stress–strain loops in a much more realistic way. However, for completeness of comparison, the behavior of isotropic hardening material, when it was cycled to same level (Fig. 105.27) and when loaded monotonically (Fig. 105.28) are also shown.

It is noted that monotonic loading curves for both perfectly plastic probabilistic model (Fig. 105.22) and linear isotropic hardening probabilistic model (Fig. 105.28) do look similar (with a noted difference of more pronounced hardening for a hardening model), but the real difference in stress–strain predictions with both probabilistic models becomes obvious in the case of cyclic loading.
Figure 105.27: Isotropic hardening probabilistic model under cyclic loading with equal loops: evolution of (a) mean and (b) standard deviation of shear stress

Figure 105.28: Isotropic hardening probabilistic model under monotonic loading: evolution of (a) mean, (b) standard deviation, and (c) mean ± standard deviation of shear stress

### 105.3.2.7 Kinematic Hardening

Expanding on elastic–plastic hardening probabilistic models, we now focus on a simple linear kinematic hardening rule based on evolution of back stress ($\alpha$). By introducing back stress ($\alpha$) to von Mises yield criteria, one can write:

$$\sqrt{J_\alpha} - k = 0$$  \hspace{1cm} (105.56)

where, $k$ is again material parameter (yield strength like) and $J_\alpha = 3/2(s_{ij} - \alpha_{ij})(s_{ij} - \alpha_{ij})$ is the $\alpha$–modified second invariant of deviatoric stress tensor ($s_{ij}$). For 1–D shear, Eq. (105.56) becomes:

$$|\sigma - \alpha| - \sigma_y = 0 \quad \text{or} \quad \sigma = \alpha \pm \sigma_y$$  \hspace{1cm} (105.57)
Hence, for kinematic hardening material, the yielding occurs at a stress of $\alpha \pm \sigma_y$, termed in the following as the equivalent yield stress. Initially, $\alpha$ is zero, and $\sigma_y$ is assumed to have a mean value of 60 kPa with a standard deviation of 20 kPa, resulting in equivalent yield stress of 60 kPa with a COV of 30%, same as the assumed yield stress for the elastic–perfectly plastic material in Section 105.3.2 and isotropic hardening material in Section 105.3.2.6. However, the same distribution of equivalent yield stress will be obtained, if one transfers the initial uncertainty from $\sigma_y$ to $\alpha$. In other words, a deterministic $\sigma_y$ of 60 kPa, and an uncertain $\alpha$ of zero mean and a standard deviation of 20 kPa will result in the same equivalent yield stress. The advantage of keeping $\sigma_y$ deterministic is that it will simplify the probabilistic addition/subtraction in Eq. (105.57), while estimating the equivalent yield stress after each incremental step of the governing FPKEs, once the back stress ($\alpha$), the internal variable for kinematic hardening material, starts evolving.

In this study, the back stress ($\alpha$) is assumed to evolve with plastic strain and hence, it would evolve probabilistically similar to probabilistic evolution of the yield strength for isotropic hardening material. Probabilistic evolution of the back stress will occur according to Eq. (105.53), with advection and diffusion coefficients given by Eq. (105.54). Shear stress evolves according to Eq. (105.46), with advection and diffusion coefficients given by Eq. (105.55). One may note that the yield strength random variable ($\Sigma_y$), appearing in Eqs. (105.54) and (105.55), is the equivalent yield strength and is given by Eq. (105.57). Fig. 105.29 shows the probabilistic evolution of shear stress in terms of mean, mode, and standard deviation, when a kinematic hardening material\textsuperscript{8}, was cycled couple of times with increasing strain loops. All other material parameters are assumed to be the same as for the elastic–perfectly plastic material in Section 105.3.2. The evolved PDFs of the back stress ($\alpha$) at the beginning and end of each branch (loading, unloading, re–loading, and re–unloading) are shown in Fig. 105.30. The evolved PDFs of equivalent yield stress (refer Eq. (105.57)) after each loading branch are shown in Fig. 105.31. Similar to the isotropic hardening case the uncertainty in (equivalent) yield strength increased as the material was cycled through, but unlike the isotropic hardening model, kinematic hardening model resulted in high kurtosis PDFs of (equivalent) yield strength. It is noted that the cyclic shear stress response of kinematic hardening material (Fig. 105.29), was more realistic than isotropic hardening material (Fig. 105.25), however, it didn’t differ much from elastic–perfectly plastic material response (Fig. 105.20). Qualitatively, those, the elastic–perfectly plastic and the kinematic hardening responses, are similar. Like the elastic–perfectly plastic material, for kinematic hardening material, the mean and mode of the evolutionary shear stress (refer Fig. 105.29) are different, although not significantly.

Similarly, when one compares response (mean and standard deviation of shear stress) for loading cycles to the same strain level, for (i) elastic–perfectly plastic, (Fig. 105.21), (ii) isotropic linear hardening

\textsuperscript{8}with non–dimensional rate of evolution of back stress with plastic strain of 10.
Figure 105.29: Kinematic hardening probabilistic model under cyclic loading with increasing loops: evolution of (a) mean, mode and (b) standard deviation of shear stress

Figure 105.30: Kinematic hardening probabilistic model under cyclic loading with increasing loops: evolved PDF of back stress at the beginning and end of (a) loading branch, (b) unloading Branch, (c) re–loading branch, and (d) re–unloading branch
Figure 105.31: Kinematic hardening probabilistic model under cyclic loading with increasing loops: evolved PDF of equivalent yield stress after (a) loading branch, (b) unloading branch, (c) re-loading branch, and (d) re-unloading branch

(Fig. 105.27), and (iii) linear kinematic hardening (Fig. 105.32), probabilistic material models, one can easily observe the qualitative similarity between elastic–perfectly plastic (i) and kinematic hardening responses (iii).

Monotonic loading cases, however, for all probabilistic material models ((i) elastic–perfectly plastic, (Fig. 105.22), (ii) isotropic linear hardening (Fig. 105.28), and (iii) linear kinematic hardening (Fig. 105.33)), are qualitatively similar, with expected differences in rate of hardening.

105.4 Stochastic Elastic-Plastic Finite Element Method
Figure 105.32: Kinematic hardening probabilistic model under cyclic loading with equal loops: evolution of (a) mean and (b) standard deviation of shear stress

Figure 105.33: Kinematic hardening probabilistic model under monotonic loading: evolution of (a) mean, (b) standard deviation, and (c) mean ± standard deviation of shear stress
Chapter 106

Large Deformation Elasto-Plasticity

(In collaboration with Dr. Zhao Cheng)
106.1 Chapter Summary and Highlights

106.2 Continuum Mechanics Preliminaries: Kinematics

106.2.1 Deformation

In modeling the material nonlinear behavior of solids, plasticity theory is applicable primarily to those bodies that can experience inelastic deformations considerably greater than the elastic deformation. If the resulting total deformation, including both translations and rotations, are small enough, we can apply small deformation theory in solving these problems. If, however strains and rotations are finite, one must resort to the theory of large deformations. In doing so, we will be using two sets of representations\(^1\), namely:

- Material coordinates in the undeformed configuration, also called Lagrangian coordinates,
- Spatial coordinates in the deformed configuration, also called Eulerian coordinates.

Figure 106.1 shows the displacement of a particle from its initial position \(X_I\) to the current position \(x_i\), defined by the deformation equation:

\[
x_i = x_i (X_1, X_2, X_3, t)
\]  

(106.1)

\[
X_I = X_I (x_1, x_2, x_3, t)
\]

(106.2)

---

\(^1\)See Malvern (1969).
The two positions are connected by the displacement $u_I$:

$$x_i = X_I + u_i \quad ; \quad X_I = x_i - u_i \quad (106.3)$$

### 106.2.2 Deformation Gradient

The **deformation gradients** are the gradients of the functions on the right-hand side of equations (106.1) and (106.2). To emphasize the difference between the material, Lagrangian setting and the spatial, Eulerian setting, we will use capital letters for the material coordinate indices and lower case letters for the spatial coordinate indices. We limit our work to the rectangular Cartesian coordinates, thus simplifying the tensor notation to the covariant indices only.

The deformation gradient is defined as the **two-point tensor** whose rectangular Cartesian components are the partial derivatives:

$$F_{kK} = \frac{\partial x_k}{\partial X_K} = x_{k,K} \quad (106.4)$$

The deformation gradient $F_{kK}$ transforms (convects) on an arbitrary infinitesimal material vector $dX_I$ at $X_I$ to associate it with a vector $dx_i$ at $x_i$:

$$dx_k = F_{kK}dX_K = \frac{\partial x_k}{\partial X_K}dX_K = x_{k,K}dX_K \quad (106.5)$$

The **spatial deformation gradients** are tensors referred to the deformed, Eulerian configuration:

$$(F_{Kk})^{-1} = \frac{\partial X_K}{\partial x_k} = X_{K,k} \quad (106.6)$$

Similarly to the deformation gradient $F_{kK}$, spatial deformation gradient $(F_{Kk})^{-1}$ operates on an arbitrary infinitesimal material vector $dx_i$ at $x_i$ to associate it with a vector $dX_I$ at $X_I$:

$$dX_K = (F_{Kk})^{-1}dx_k = \frac{\partial X_K}{\partial x_k}dx_k = X_{K,k}dx_k \quad (106.7)$$

The spatial deformation gradient $(F_{Kk})^{-1}$ at $x_i$ is the inverse to the two-point tensor $F_{kK}$ at $X_I$:

$$F_{iJ}(F_{Jk})^{-1} = \delta_{ik} \quad \text{and} \quad (F_{ij})^{-1}F_{jK} = \delta_{IK} \quad (106.8)$$

The **Jacobian** of the mapping (106.4) can be represented as:

$$J = \det (F_{kK}) = \frac{1}{6} e_{ijk}e_{PQR}F_{iP}F_{jQ}F_{kR} \quad (106.9)$$
The relative deformation gradient \( f_{km} \) is the gradient for the relative motion function:

\[
\xi = \chi_t (x_i, \tau)
\]  
(106.10)

and is defined as:

\[
f_{km} = \xi_{k,m} \equiv \frac{\partial \xi_k}{\partial x_m}
\]  
(106.11)

If the fixed reference position \( X_i \), the current position \( x_i \) and the variable position \( \xi_i \) are all referred to the rectangular Cartesian coordinate system, the chain rule of differentiation yields:

\[
\frac{\partial \xi_k}{\partial X_I} = \frac{\partial \xi_k}{\partial x_m} \frac{\partial x_m}{\partial X_I} \quad \text{or} \quad F_{kI} = f_{km} F_{mI}
\]  
(106.12)

The polar decomposition theorem permits the unique representation\(^2\):

\[
F_{ij} = R_{ik} U_{kj} = v_{ik} R_{kj}
\]  
(106.13)

where \( U_{kj}, v_{ik} \) are positive definite symmetric tensors, called right stretch tensors and left stretch tensors, respectively, and \( R_{kj} \) is an orthogonal tensor such that:

\[
R_{ik} R_{jk} = \delta_{ij} \quad \text{and also} \quad R_{ki} R_{kj} = \delta_{ij}
\]  
(106.14)

Equation (106.13), as well as Figure 106.2.2 demonstrate that the motion and deformation of an infinitesimal volume element at \( X_i \) consist of consecutive applications of:

- a stretch by \( U_{kj} \),
- a rigid body rotation by \( R_{ik} \),
- a rigid body translation to \( x_i \)

or alternatively:

- a rigid body translation to \( x_i \)
- a rigid body rotation by \( R_{kj} \),
- a stretch by \( v_{ik} \),

\(^2\)referring \( x_i \) and \( X_i \) to the same reference axes and using lower case indices for both. This reference to the same coordinate system will be applied only for the polar decomposition example presented here.
106.2.3 Strain Tensors, Deformation Tensors and Stretch

The strain tensors $E_{IJ}$ and $e_{ij}$ are defined so that they give the change in the square length of the material vector $dX_I$. For the Lagrangian formulation we write:

$$(ds)^2 - (dS)^2 = 2dX_I E_{IJ} dX_J$$

(106.15)

and for the Eulerian formulation:

$$(ds)^2 - (dS)^2 = 2dx_i e_{ij} dx_j$$

(106.16)

The deformation tensors $C_{IJ}$ and $c_{ij}$ are connecting the squared lengths in Lagrangian and Eulerian configurations. The Green deformation tensor$^3$ $C_{IJ}$, referred to the undeformed configuration, gives the new squared length $(ds)^2$ of the element into which the given element $dX_I$ is deformed:

$$(ds)^2 = dX_I C_{IJ} dX_J$$

(106.17)

The Cauchy deformation tensor $c_{ij}$, sometimes also denoted as$^4$ $(b_{ij})^{-1}$, gives the initial squared length $(dS)^2$ of an element $dx_i$ identified in the deformed configuration:

$$(dS)^2 = dx_i c_{ij} dx_j$$

(106.18)

---

$^3$Also called right Cauchy–Green tensor.

$^4$Another name for $b_{ij}$ is Finger deformation tensor or left Cauchy–Green tensor.
Substituting equation (106.17) into (106.15) yield:

$$2E_{IJ} = C_{IJ} - \delta_{IJ}$$  \hspace{1cm} (106.19)

and similarly, substituting equation (106.18) into (106.16) we obtain:

$$2e_{ij} = \delta_{ij} - c_{ij}$$  \hspace{1cm} (106.20)

By using equation (106.5) we can express \((ds)^2\) as:

$$\left(\frac{ds}{dX}\right)^2 = dx_k dx_k = (F_{kI} dx_I)(F_{kJ} dx_J) = (x_{k, I} dx_I)(x_{k, J} dx_J) = dX_I (F_{kI} F_{kJ}) dX_K = dX_I C_{IJ} dX_K$$  \hspace{1cm} (106.21)

so we have obtained the connection between the deformation tensor \(C_{IJ}\) and the deformation gradient \(F_{kI}\) in the form:

$$C_{IJ} = (F_{kI} F_{kJ}) = x_{k, I} dx_I x_{k, J} dx_J$$  \hspace{1cm} (106.22)

Similarly, by using equation (106.7) and the expression for \((dS)^2\) we can establish the connection between the deformation tensor \(c_{ij}\) and the deformation gradient \(F_{Ki}\) as:

$$\left(\frac{dS}{dx}\right)^2 = dS_K dX_K = (F_{Ki} dx_i) (F_{Kj} dx_j) = (X_{K, i} dx_i) (X_{K, j} dx_j) = dx_i (F_{Ki} F_{Kj}) dx_k = dx_i c_{ij} dx_k \Rightarrow$$

$$c_{ij} = (F_{Ki})^{-1} F_{Kj}^{-1}$$  \hspace{1cm} (106.23)

The expressions for the strain tensors in Lagrangian and Eulerian description\(^5\) is obtained from equations (106.19) and (106.20):

\[\text{L: } E_{IJ} = \frac{1}{2} \left( (F_{kI} F_{kJ}) - \delta_{IJ} \right) \quad ; \quad \text{E: } e_{ij} = \frac{1}{2} \left( \delta_{ij} - (F_{Ki})^{-1} (F_{Kj})^{-1} \right) \]  \hspace{1cm} (106.24)

If one starts from the displacement equation (106.3), referenced to the same axes for both \(X_I\) and \(x_i\)

$$x_I = X_I + u_I \quad ; \quad X_I = x_I - u_I$$

the general expression for the Lagrangian strain tensor \(E_{IJ}\) in terms of displacements is:

$$E_{IJ} = \frac{1}{2} \left( (F_{KI} F_{KJ}) - \delta_{IJ} \right) = \frac{1}{2} \left( (\delta_{KI} + u_{K, I}) (\delta_{KJ} + u_{K, J}) - \delta_{IJ} \right) = \frac{1}{2} \left( \delta_{KI} \delta_{KJ} + \delta_{KI} u_{K, J} + u_{K, I} \delta_{KJ} + u_{K, I} u_{K, J} - \delta_{IJ} \right) = \frac{1}{2} \left( \delta_{IJ} + u_{I, J} + u_{J, I} + u_{K, I} u_{K, J} - \delta_{IJ} \right) = \frac{1}{2} \left( u_{I, J} + u_{J, I} + u_{K, I} u_{K, J} \right)$$  \hspace{1cm} (106.25)

\(^5\)Lagrangian format will be denoted by \(\text{L:}\) while Eulerian format by \(\text{E:}\).
Similarly, the general expression for the Eulerian strain tensor $e_{ij}$ in terms of displacements is:

$$e_{ij} = \frac{1}{2} \left( \delta_{ij} - (F_{ki})^{-1} (F_{kj})^{-1} \right) = \frac{1}{2} \left( \delta_{ij} - (\delta_{ki} - u_{ki}) (\delta_{kj} - u_{kj}) \right) = \frac{1}{2} \left( \delta_{ij} - \delta_{ki} \delta_{kj} + \delta_{ki} u_{kj} + \delta_{kj} u_{ki} - u_{ki} u_{kj} \right) = \frac{1}{2} \left( \delta_{ij} + u_{i,j} + u_{j,i} - u_{ki} u_{kj} \right) = \frac{1}{2} \left( u_{i,j} + u_{j,i} - u_{ki} u_{kj} \right)$$  \hspace{1cm} (106.26)

It is worthwhile noting that equations (106.25) and (106.26) represent the complete finite strain tensor. They involve only linear and quadratic terms in the components of displacement gradients.

The stretch is a measure of extension of an infinitesimal element and is a function of direction of an element, in either deformed or undeformed configuration. By denoting $N_I$ a unit vector in the undeformed configuration and $n_i$ a unit vector in the deformed configuration, we denote material stretch as $\Lambda^{(N)}_{n}$ of those elements with initial direction $N_i$ and spatial stretch $\lambda_{(n)}$ of those elements with initial direction $n_i$. By dividing equations (106.15) and (106.16) by $(ds)^2$ and $(dS)^2$ respectively and by using:

$$N_I = \frac{dX_I}{dS} \quad \text{and} \quad n_i = \frac{dx_i}{ds}$$  \hspace{1cm} (106.27)

we obtain the Cartesian form of stretch in the Lagrangian and Eulerian descriptions:

\[
\textbf{L:} \quad \Lambda^{(N)}_{N} = \frac{dX_I}{dS} C_{IJ} \frac{dX_J}{dS} \quad \text{and} \quad \textbf{E:} \quad \lambda^{(n)} = \frac{dx_i}{ds} c_{ij} \frac{dx_j}{ds}  \hspace{1cm} (106.28)
\]

General strain tensors can be defined by considering a scale function \cite{Hill,1978} for the stretch. Scale function is any smooth, monotonic function of stretch $f(\lambda)$ such that:

$$f(\lambda) ; \lambda \in [0, \infty) \text{ subject to } f(1) = 0, f'(1) = 1$$  \hspace{1cm} (106.29)

Scale function is often taken in the form $(\lambda^{2m} - 1)/2m$, where $m$ may have any value. If we choose $m$ to be an integer, the corresponding strain tensor is:

$$E_{IJ} = \frac{(U_{IJ}^{2m} - \delta_{IJ})}{2m} \quad \text{where} \quad F_{IJ} = R_{IK} U_{KJ} = v_{IK} R_{KJ}  \hspace{1cm} (106.30)$$

Table 106.1 shows different Lagrangian strain measures obtained for a particular choice of parameter $m$.

In the Eulerian setting, generalized strain tensor is defined as

$$e_{ij} = \frac{(\delta_{ij} - v_{ij}^{2m})}{2m} ; \quad F_{IJ} = R_{IK} U_{KJ} = v_{IK} R_{KJ}$$  \hspace{1cm} (106.31)
Table 106.1: Different Lagrangian strain measures.

<table>
<thead>
<tr>
<th>Strain measure name</th>
<th>parameter $m$</th>
<th>expression for $E_{IJ}^m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green–Lagrange</td>
<td>1</td>
<td>$E_{IJ}^{GL} = (U_{IJ}^2 - \delta_{IJ}) / 2$</td>
</tr>
<tr>
<td>Almansi</td>
<td>-1</td>
<td>$E_{IJ}^A = (\delta_{IJ} - U_{IJ}^{-2}) / 2$</td>
</tr>
<tr>
<td>Biot</td>
<td>1/2</td>
<td>$E_{IJ}^B = (U_{IJ} - \delta_{IJ})$</td>
</tr>
<tr>
<td>Hencky</td>
<td>0</td>
<td>$E_{IJ}^H = \ln (U_{IJ})$</td>
</tr>
</tbody>
</table>

106.2.4 Rate of Deformation Tensor

The rate of deformation tensor$^6$ describes the tangent motion in terms of velocity components $v_i = dx_i/dt$. The spatial coordinates are:

$$v_i = v_i(x_1, x_2, x_3, t)$$

(106.32)

Figure 106.3: Relative velocity $dv_i$ of particle $Q$ at point $q$ relative to particle $P$ at point $p$.

In Figure 106.3 the dashed lines represents the trajectories of particles $P$ and $Q$. The velocity vectors $v_i$ at $p$ and $v_i + dv_i$ at $q$ are tangent to the two trajectories. The relative velocity components $dv_i$ of particle at $q$ relative to the particle at $p$ are given by:

$$dv_k = \frac{\partial v_k}{\partial x_m} dx_m = v_{k,m} dx_m = L_{km} dx_m$$

(106.33)

$^6$Also called stretch tensor or velocity strain.
The spatial gradient of the velocity \( L_{km} \) can be decomposed as the sum of the symmetric, rate of deformation tensor \( D_{km} \), and a skew symmetric spin tensor \( W_{km} \) as follows:

\[
L_{km} = \frac{1}{2} (L_{km} + L_{mk}) + \frac{1}{2} (L_{km} - L_{mk}) = D_{km} + W_{km}
\]  

(106.34)

where:

\[
D_{km} = \frac{1}{2} (L_{km} + L_{mk}) = D_{mk} \quad \text{and} \quad W_{km} = \frac{1}{2} (L_{km} - L_{mk}) = -W_{mk}
\]  

(106.35)

An alternate way of deriving the rate of deformation tensor goes as follows. The rate of change of squared length \((ds)^2\) is given as:

\[
\frac{d(ds)^2}{dt} = 2 \frac{d(ds)}{dt} ds
\]  

(106.36)

since \((ds)^2 = dx_k dx_k\) it follows:

\[
\frac{d(ds)}{dt} = 2 \frac{d(dx_k)}{dt} dx_k
\]  

(106.37)

and with \(dx_k = (\partial x_k / \partial X_m) dX_m\) it follows:

\[
\frac{d(dx_k)}{dt} = \frac{d}{dt} \left( \frac{\partial x_k}{\partial X_m} \right) dX_m + \frac{d(dX_m)}{dt} \frac{\partial x_k}{\partial X_m} = \frac{d}{dt} \left( \frac{\partial x_k}{\partial X_m} \right) dX_m
\]  

(106.38)

since \(d(dX_m)/dt \equiv 0\), because the initial relative position vector \(dX_m\) does not change with time. By interchanging the order of differentiation we get:

\[
\frac{d(dx_k)}{dt} = dv_k = \frac{d}{dt} \left( \frac{\partial x_k}{\partial X_m} \right) dX_m = \frac{\partial v_k}{\partial X_m} dX_m \quad \text{where} \quad v_k = \frac{\partial x_k}{\partial X_m}
\]  

(106.39)

From equation (106.33) \(dv_k = L_{km} dx_m\) and equation (106.39) it follows that:

\[
\frac{\partial v_k}{\partial X_m} dX_m = L_{km} dx_m \quad \Rightarrow \quad \frac{d(dx_k)}{dt} = dv_k = L_{km} dx_m = v_{k,m} dx_m
\]  

(106.40)

and then the equation (106.37) becomes:

\[
\frac{d(ds)}{dt} = 2 \frac{d(dx_k)}{dt} dx_k = 2 dx_k v_{k,m} dx_m = 2 dx_k L_{km} dx_m dx_k = 2 dx_k D_{km} dx_m dx_k = 2 dx_k D_{km} dx_m dx_k
\]  

(106.41)

since \(dx_k dx_m = dx_m dx_k\) and \(W_{km}\) is skew symmetric such that \(W_{km} = -W_{mk}\). Finally we obtain:

\[
\frac{d(ds)}{dt} = 2 dx_k D_{km} dx_m
\]  

(106.42)

and thus it follows that the rate of change of the squared length \((ds)^2\) of the material instantaneously occupying any infinitesimal relative position \(dx_k\) at point \(p\) is determined by the tensor \(D_{km}\) at point \(p\).
In order to compare the strain rate to the rate of deformation, we differentiate equation (106.15) with respect to time:

\[
\frac{d}{dt} \left( (ds)^2 - (dS)^2 \right) = 2 \frac{d}{dt} (dX_I E_{IJ} dX_J) = \\
= \frac{d}{dt} (ds)^2 = 2dX_I \frac{d}{dt} (E_{IJ}) dX_J
\]

(106.43)

since \((dS)^2\) and \(dX_I\) are constant through time. From the equations (106.42) and (106.43) it follows that:

\[
\frac{d}{dt} (ds)^2 = 2dx_k D_{km} dx_m = 2 (dX_I F_{Ik}) D_{km} (F_{mJ} dX_J) = 2dX_I (F_{Ik} D_{km} F_{mJ}) dX_J
\]

(106.44)

and from equations (106.43) and (106.44) it follows that:

\[
\frac{dE_{IJ}}{dt} = F_{Ik} D_{km} F_{mJ}
\]

(106.45)

or inversely:

\[
D_{km} = (F_{Ik})^{-1} \frac{dE_{IJ}}{dt} (F_{mJ})^{-1}
\]

(106.46)

To obtain the rate of change of the deformation gradient we start from equations (106.4) and differentiate it with respect to time:

\[
\frac{dF_{kK}}{dt} = d \left( \frac{\partial x_k}{\partial x_K} \right) = \frac{\partial v_k}{\partial X_K} = \frac{\partial v_k}{\partial x_m} \frac{\partial x_m}{\partial X_K} = v_{k,m} x_{m,K} = \frac{dx_{k,K}}{dt} = \\
= L_{km} F_{mK} = \dot{F}_{kK}
\]

(106.47)

or inversely:

\[
v_{k,m} = \frac{dx_{k,K}}{dt} X_{K,m} = \frac{dF_{kK}}{dt} (F_{km})^{-1} = \\
= \dot{F}_{kK} (F_{km})^{-1} = L_{km}
\]

(106.48)
106.3 Constitutive Relations: Hyperelasticity

106.3.1 Introduction

A material is called *hyperelastic* or *Green elastic*, if there exists an *elastic potential function* \( W \), also called the *strain energy function per unit volume of the undeformed configuration*, which represents a scalar function of strain of deformation tensors, whose derivatives with respect to a strain component determines the corresponding stress component. The most general form of the elastic potential function, is described in equation 106.49, with restriction to pure mechanical theory, by using the *axiom of locality* and the *axiom of entropy production*\(^7\):

\[
W = W(X_K, F_{kK})
\]  
(106.49)

By using the *axiom of material frame indifference*\(^8\), we conclude that \( W \) depends only on \( X_K \) and \( C_{IJ} \), that is:

\[
W = W(X_K, C_{IJ}) \quad \text{or} \quad W = W(X_K, c_{ij})
\]  
(106.50)

By assuming hyperelastic response, the following are the constitutive equations for the material stress tensors:

- **2. Piola–Kirchhoff stress tensor:**
  \[
  S_{IJ} = 2 \frac{\partial W}{\partial C_{IJ}}
  \]  
  (106.51)

- **Mandel stress tensor:**
  \[
  T_{IJ} = C_{IK}S_{KJ} = 2C_{IK} \frac{\partial W}{\partial C_{KJ}}
  \]  
  (106.52)

- **1. Piola–Kirchhoff stress tensor**
  \[
  P_{iJ} = S_{IJ}(F_{it})^t = 2 \frac{\partial W}{\partial C_{IJ}}(F_{it})^t
  \]  
  (106.53)

and the spatial, Kirchhoff stress tensor is defined as:

---

\(^7\)See Marsden and Hughes (1983) pp. 190.

\(^8\)See Marsden and Hughes (1983) pp. 194.
• Kirchhoff stress tensor

\[
\tau_{ij} = 2 \frac{\partial W}{\partial b_{ij}} = 2 F_{iA}(F_{jB})^t \frac{\partial W}{\partial C_{AB}} = F_{iA}(F_{jB})^t S_{AB}
\] (106.54)

Material tangent stiffness relation is defined from:

\[
dS_{IJ} = 2 \frac{\partial^2 W}{\partial C_{IJ} \partial C_{KL}} dC_{KL} = \frac{1}{2} \mathcal{L}_{IJKL} dC_{KL}
\] (106.55)

where

\[
\mathcal{L}_{IJKL} = 4 \frac{\partial^2 W}{\partial C_{IJ} \partial C_{KL}}
\] (106.56)

The spatial tangent stiffness tensor \( \mathcal{E}_{ijkl} \) is obtained by the following push–forward operation with the deformation gradient:

\[
\mathcal{E}_{ijkl} = F_{il} F_{jJ}(F_{kK})^t (F_{lL})^t \mathcal{L}_{IJKL}
\] (106.57)

106.3.2 Isotropic Hyperelasticity

In the case of material isotropy, the strain energy function \( W(X_K, C_{IJ}) \) belongs to the class of isotropic, invariant scalar functions. It satisfies the relation:

\[
W(X_K, C_{KL}) = W(X_K, Q_{KI} C_{IJ} (Q_{JL})^t)
\] (106.58)

where \( Q_{KI} \) is the proper orthogonal transformation. If we choose \( Q_{KI} = R_{KI} \), where \( R_{KI} \) is the orthogonal rotation transformation, defined by the polar decomposition theorem in equation (106.13), then:

\[
W(X_K, C_{KL}) = W(X_K, U_{KL}) = W(X_K, v_{kl})
\] (106.59)
Right and left stretch tensors, $U_{KL}$, $v_{kl}$ have the same principal values $\lambda_i$; $i = 1, 3$ so the strain energy function $W$ can be represented in terms of principal stretches, or similarly in terms of principal invariants of deformation tensor:

$$W = W(X_K, \lambda_1, \lambda_2, \lambda_3) = W(X_K, I_1, I_2, I_3)$$

(106.60)

where:

$$I_1 \overset{\text{def}}{=} C_{II}$$

$$I_2 \overset{\text{def}}{=} \frac{1}{2} (I_1^2 - C_{JJ} C_{II})$$

$$I_3 \overset{\text{def}}{=} \text{det} (C_{IJ}) = \frac{1}{6} e_{IJK} e_{PQR} C_{IP} C_{JQ} C_{KR} = J^2$$

(106.61)

Left and right Cauchy–Green tensors were defined by equations (106.22) and (106.23), respectively as:

$$C_{IJ} = (F_{kI})^t F_{kJ}; \quad (e^{-1})_{ij} = b_{ij} = F_{iK} (F_j K)^t$$

(106.62)

The spectral decomposition theorem for symmetric positive definite tensors states that:

$$C_{IJ} = \lambda_A^2 \left( N_{I}^{(A)} N_{J}^{(A)} \right)_A$$

where $A = 1, 3$

(106.63)

and $N_I$ are the eigenvectors of $C_{IJ}$. Values $\lambda_A^2$ are the roots of the characteristic polynomial

$$P(\lambda_A^2) \overset{\text{def}}{=} -\lambda_A^6 + I_1 \lambda_A^4 - I_2 \lambda_A^2 + I_3 = 0$$

(106.64)

It should be noted that no summation is implied over indices in parenthesis.

The mapping of the eigenvectors can be deduced from equation (106.5) and is given by

$$\lambda_{(A)} n_i^{(A)} = F_{iJ} N_{J}^{(A)}$$

(106.65)

where $\|n_i^{(A)}\| = 1$. The spectral decomposition of $F_{iJ}$, $R_{iJ}$ and $b_{ij}$ is then given by

$$F_{iJ} = \lambda_A \left( n_i^{(A)} N_{J}^{(A)} \right)_A$$

(106.66)

$$R_{iJ} = \sum_{A=1}^{3} n_i^{(A)} N_{J}^{(A)}$$

(106.67)

---

9Principal stretches.


11Cauchy–Green tensor $C_{IJ}$ for example.

12So that $\|N_I\| = 1$.

13For example, in the present case $N_I^{(A)}$ is the $A$th eigenvector with members $N_1^{(A)}$, $N_2^{(A)}$ and $N_3^{(A)}$, so that the actual equation $C_{IJ} = \lambda_A^2 \left( N_I^{(A)} N_J^{(A)} \right)_A$ can also be written as $C_{IJ} = \sum_{A=1}^{3} \lambda_A^2 (N_I^{(A)} N_J^{(A)})$. In order to follow the consistency of indicial notation in this work, we shall make an effort to represent all the tensorial equations in indicial form.
\[ b_{ij} = \lambda_A^2 \left( n_i^{(A)} n_j^{(A)} \right)_A \]  

(106.68)

Spectral decomposition from equation (106.63) is valid for the case of non–equal principal stretches, i.e. \( \lambda_1 \neq \lambda_2 \neq \lambda_3 \). If two or all three principal stretches are equal, we shall introduce a small perturbation to the numerical values for principal stretches in order to make them distinct. The case of two or all three values of principal stretches being equal is theoretically possible and results for example from standard triaxial tests or isotropic compression tests. However, we are never certain about equivalence of two numerical numbers, because of the finite precision arithmetics involved in calculation of these numbers. From the numerical point of view, two number are equal if the difference between them is smaller than the machine precision \((\text{macheps})\) specific to the computer platform on which computations are performed. Our perturbation will be a function of the \(\text{macheps}\).

The characteristic polynomial \(P(\lambda_A^2)\) from equation (106.64) can be solved\(^{14}\) for \(\lambda_A\):

\[ \lambda_A = \frac{1}{\sqrt{3}} \sqrt{I_1 + 2 \sqrt{I_1^2 - 3I_2} \cos \left( \frac{\Theta + 2\pi A}{3} \right)} \]  

(106.69)

where

\[ \Theta = \arccos \frac{2I_3^3 - 9I_1I_2 + 27I_3}{2\sqrt{(I_1^2 - 3I_2)^3}} \]  

(106.70)

Recently, Ting (1985) and Morman (1986) have used Serrin’s representation theorem in order to devise a useful representation for generalized strain tensors\(^{15}\) \(\varepsilon_{IJ}\) and \(e_{ij}\) through \(C_{IJ}^m\) and \(b_{ij}^m\). Morman (1986) has shown that \(b_{ij}^m\) can be stated as

\[ b_{ij}^m = \lambda_A^{2m} \left( \frac{(b^2)_{ij} - \left( I_1 - \lambda_A^2 \right) b_{ij} + I_3 \lambda_A^{-2} \delta_{ij}}{2\lambda_A^4 - I_1 \lambda_A^2 + I_3 \lambda_A^{-2}} \right)_A \]  

(106.71)

By comparing equations (106.71) and (106.68) it follows that the Eulerian eigendiad \(n_i^{(A)} n_j^{(A)}\) can be written as

\[ n_i^{(A)} n_j^{(A)} = \frac{(b^2)_{ij} - \left( I_1 - \lambda_A^2 \right) b_{ij} + I_3 \lambda_A^{-2} \delta_{ij}}{2\lambda_A^4 - I_1 \lambda_A^2 + I_3 \lambda_A^{-2}} \]  

(106.72)

The Lagrangian eigendiad \(N_i^{(A)} N_j^{(A)}\), from equation (106.63), can be derived, if one substitutes mapping of the eigenvectors, (106.65), into equation (106.72) to get:

\[ N_i^{(A)} N_j^{(A)} = \lambda_A^2 \frac{C_{IJ} - \left( I_1 - \lambda_A^2 \right) \delta_{IJ} + I_3 \lambda_A^{-2} (C^{-1})_{IJ}}{2\lambda_A^4 - I_1 \lambda_A^2 + I_3 \lambda_A^{-2}} \]  

(106.73)

\(^{14}\)See also Morman (1986) and Schellekens and Schellekens and Parisch (1994).

\(^{15}\)Defined by equations (106.30) and (106.31).
where it was used that:
\[
C_{IJ} = (F_{II})^{-1} (b^2)_{ij} (F_{JJ})^{-t} \tag{106.74}
\]
\[
\delta_{IJ} = (F_{II})^{-1} b_{ij} (F_{JJ})^{-t} \tag{106.75}
\]
\[
(C^{-1})_{IJ} = (F_{II})^{-1} \delta_{ij} (F_{JJ})^{-t} \tag{106.76}
\]

It should be noted that the denominator in equations (106.72) and (106.73) can be written as:
\[
2\lambda_1^4 - I_1\lambda_2^2 + I_3\lambda_3^2 = \left(\lambda_1^2 - \lambda_2^2\right) \left(\lambda_2^2 - \lambda_3^2\right) \quad \text{def} \quad D_A \tag{106.77}
\]
where indices \(A, B, C\) are cyclic permutations of \(1, 2, 3\). It follows directly from the definition of \(D_A\) in equation (106.77) that \(\lambda_1 \neq \lambda_2 \neq \lambda_3 \Rightarrow D_A \neq 0\) for equations (106.72) and (106.73) to be valid.

Similarly to equations (106.63) and (106.68) we can obtain:
\[
(C^{-1})_{IJ} = \lambda_A^{-2} \left(N_I^{(A)} N_J^{(A)}\right)_A \tag{106.78}
\]
\[
(b^{-1})_{ij} = \lambda_A^{-2} \left(n_i^{(A)} n_j^{(A)}\right)_A \tag{106.79}
\]

### 106.3.3 Volumetric–Isochoric Decomposition of Deformation

It proves useful to separate deformation in volumetric and isochoric parts by a multiplicative split of a deformation gradient as
\[
F_{II} = \tilde{F}_{\beta I} \quad \text{v} \quad \text{F}_{\beta I} = F_{II} J^{-\frac{1}{3}} \quad ; \quad \text{v} \quad \text{F}_{\beta I} = J^{\frac{1}{3}} \delta_{\beta I} \tag{106.80}
\]
where \(x_\beta\) represents an intermediate configuration such that deformation \(X_I \rightarrow x_\beta\) is purely volumetric and \(x_\beta \rightarrow x_i\) is purely isochoric. It also follows from equation (106.80) that \(\tilde{F}_{\beta I}\) and \(F_{II}\) have the same eigenvectors.

The isochoric part of the Green deformation tensor \(C_{IJ}\), defined in equation (106.63) can be defined as
\[
\tilde{C}_{IJ} = J^{-\frac{1}{3}} C_{IJ} = \lambda_A^2 \left(N_I^{(A)} N_J^{(A)}\right)_A \tag{106.81}
\]
while the isochoric part of the Finger deformation tensor \(b_{ij}\) can be defined similarly as
\[
\tilde{b}_{ij} = J^{-\frac{2}{3}} b_{ij} = \lambda_A^2 \left(n_i^{(A)} n_j^{(A)}\right)_A \tag{106.82}
\]
where the isochoric principal stretches are defined as
\[
\tilde{\lambda}_A = J^{-\frac{1}{3}} \lambda_A = (\lambda_1 \lambda_2 \lambda_3)^{-\frac{1}{3}} \lambda_A \tag{106.83}
\]

The free energy \(W\) is then decomposed additively as:
\[
W(X_K, \lambda_{(A)}) = \text{iso} W(X_K, \tilde{\lambda}_{(A)}) + \text{v} \quad W(X_K, J) \tag{106.84}
\]
106.3.4 Simo–Serrin’s Formulation

In Section (106.3.2) we have presented the most general form of the isotropic strain energy function $W$ in terms of of principal stretches:

$$ W = W (X_K, \lambda_1, \lambda_2, \lambda_3, ) $$

(106.85)

It was also shown in Section (106.3.1) that it is necessary to calculate the gradient $\partial W / \partial C_{IJ}$ in order to obtain $2$. Piola–Kirchhoff stress tensor $S_{IJ}$ and accordingly other stress measures. Likewise, it was shown that the material tangent stiffness tensor $L_{IJKL}$ (as well as the spatial tangent stiffness tensor $E_{ijkl}$) requires second order derivatives of strain energy function $\partial^2 W / (\partial C_{IJ} \partial C_{KL})$. In order to obtain these quantities we introduce\(^\text{16}\) a second order tensor $M^{(A)}_{IJ}$

$$ M^{(A)}_{IJ} \overset{\text{def}}{=} \lambda^{-2}_{(A)} N^{(A)}_I N^{(A)}_J $$

(106.86)

$$ = (F_{II})^{-1} \left( n^{(A)}_i n^{(A)}_j \right) (F_{JJ})^{-t} $$

$$ = \frac{1}{D_{(A)}} \left( C_{IJ} - \left( I_1 - \lambda^2_{(A)} \right) \delta_{IJ} \right) \delta_{IJ} + I_3 \lambda^{-2}_{(A)} (C^{-1})_{IJ} $$

from (106.73)

where $D_{(A)}$ was defined by equation (106.77). With $M^{(A)}_{IJ}$ defined by equation (106.86), we get from equation (106.63) that:

$$ C_{IJ} = \lambda^4_A \left( M^{(A)}_{IJ} \right)_A $$

(106.87)

and also from equation (106.78) it follows that:

$$ (C^{-1})_{IJ} = M^{(1)}_{IJ} + M^{(2)}_{IJ} + M^{(3)}_{IJ} $$

(106.88)

\(^\text{16}\)See Runesson (1996).
It can also be concluded that:

$$\delta_{IJ} = \lambda_2^2 M_{IJ}^{(1)} + \lambda_2^2 M_{IJ}^{(2)} + \lambda_2^2 M_{IJ}^{(3)} = \lambda_2^2 \left( M_{IJ}^{(A)} \right)_A$$  \hspace{1cm} (106.89)

since, from the orthogonal properties of eigenvectors

$$\delta_{IJ} = \sum_{A=1}^{3} N_{I}^{(A)} N_{J}^{(A)} = \left( N_{I}^{(A)} \right)_A \left( N_{J}^{(A)} \right)_A$$  \hspace{1cm} (106.90)

We are now in a position to define the Simo–Serrin fourth order tensor $M_{IJKL}$ as:

$$M_{IJKL}^{(A)} \overset{\text{def}}{=} \frac{\partial M_{IJ}^{(A)}}{\partial C_{KL}} = \frac{1}{D_{(A)}} \left( I_{IJKL} - \delta_{KL} \delta_{IJ} + \lambda_2^2 \left( \delta_{IJ} M_{KL}^{(A)} + M_{IJ}^{(A)} \delta_{KL} \right) + I_3 \lambda_2^2 \left( (C^{-1})_{IJ} (C^{-1})_{KL} + \frac{1}{2} \left( (C^{-1})_{IK} (C^{-1})_{JL} + (C^{-1})_{IL} (C^{-1})_{JK} \right) \right) - \lambda_2^2 I_3 \left( (C^{-1})_{IJ} M_{KL}^{(A)} + M_{IJ}^{(A)} (C^{-1})_{KL} \right) - D'_{(A)} M_{IJ}^{(A)} M_{KL}^{(A)} \right)$$  \hspace{1cm} (106.91)

Complete derivation of $M_{IJKL}$ is given in Appendix (704.2).

### 106.3.5 Stress Measures

In Section (106.3.1) we have defined various stress measures in terms of derivatives of the free energy function $W$. With the free energy function decomposition, as defined in equation (106.84) we can appropriately decompose all the previously defined stress measures:

- 2. Piola–Kirchhoff stress tensor:
  
  $$S_{IJ} = 2 \frac{\partial W}{\partial C_{IJ}} = 2 \frac{\partial W}{\partial C_{IJ}} + 2 \frac{\partial W}{\partial C_{IJ}} = \text{iso} S_{IJ} + \text{vol} S_{IJ}$$  \hspace{1cm} (106.92)

- Mandel stress tensor:
  
  $$T_{IJ} = C_{IK} S_{KJ} = 2 C_{IK} \frac{\partial W}{\partial C_{KJ}} = 2 C_{IK} \frac{\partial W}{\partial C_{KJ}} + 2 C_{IK} \frac{\partial W}{\partial C_{KJ}} = \text{iso} T_{IJ} + \text{vol} T_{IJ}$$  \hspace{1cm} (106.93)
1. Piola–Kirchhoff stress tensor

\[ P_{iJ} = S_{IJ}(F_{ii})^t = 2 \frac{\partial W}{\partial C_{IJ}}(F_{ii})^t = 2 \frac{\partial^{\text{iso}} W}{\partial C_{IJ}}(F_{ii})^t + 2 \frac{\partial^{\text{vol}} W}{\partial C_{IJ}}(F_{ii})^t \]

\[ = \text{iso} P_{iJ} + \text{vol} P_{iJ} \]  

(106.94)

Kirchhoff stress tensor

\[ \tau_{ab} = 2 \frac{\partial W}{\partial e_{ij}} = F_{aI}(F_{bJ})^t S_{IJ} = 2 F_{aI}(F_{bJ})^t \frac{\partial^{\text{iso}} W}{\partial C_{IJ}} + 2 F_{aI}(F_{bJ})^t \frac{\partial^{\text{vol}} W}{\partial C_{IJ}} \]

\[ = F_{aI}(F_{bJ})^t \text{iso} S_{IJ} + F_{aI}(F_{bJ})^t \text{vol} S_{IJ} \]

\[ = \text{iso} \tau_{ab} + \text{vol} \tau_{ab} \]  

(106.95)

The derivative of the volumetric part of the free energy function is

\[ \frac{\partial^{\text{vol}} W(J)}{\partial C_{IJ}} = \frac{\partial^{\text{vol}} W(J)}{\partial J} \frac{\partial J}{\partial C_{IJ}} = \frac{1}{2} \frac{\partial^{\text{vol}} W(J)}{\partial J} \quad (C^{-1})_{IJ} \]  

(106.96)

where equation (704.9) was used, while the derivative of the isochoric part of the free energy function yields

\[ \frac{\partial^{\text{iso}} W(\tilde{\lambda}(A))}{\partial C_{IJ}} = \frac{\partial^{\text{iso}} W(\tilde{\lambda}(A))}{\partial \tilde{\lambda}(A)} \frac{\partial \tilde{\lambda}(A)}{\partial C_{IJ}} = \frac{1}{2} \frac{\partial^{\text{iso}} W(\tilde{\lambda}(A))}{\partial \tilde{\lambda}(A)} \tilde{\lambda}(A) + \frac{\partial^{\text{iso}} W(\tilde{\lambda}(A))}{\partial \tilde{\lambda}(A)} \tilde{\lambda}(A) \]  

(106.97)

where equation (704.7) was used and \( w_A \) is derived in Appendix 704.5 as:

\[ w_A = \frac{\partial^{\text{iso}} W(\lambda(A))}{\partial \tilde{\lambda}(A)} \frac{\partial \tilde{\lambda}(A)}{\partial \lambda(A)} \tilde{\lambda}(A) = -\frac{1}{3} \frac{\partial^{\text{iso}} W(\tilde{\lambda}(A))}{\partial \tilde{\lambda}(A)} \tilde{\lambda}(A) + \frac{\partial^{\text{iso}} W(\tilde{\lambda}(A))}{\partial \tilde{\lambda}(A)} \tilde{\lambda}(A) \]  

(106.98)

The decomposed 2. Piola–Kirchhoff stress tensor is

\[ S_{IJ} = \text{vol} S_{IJ} + \text{iso} S_{IJ} \]

\[ = \frac{\partial^{\text{vol}} W(J)}{\partial J} J (C^{-1})_{IJ} + w_A (M_{IJ}^{(A)})_A \]  

(106.99)

The derivative of the free energy is then:

\[ \frac{\partial W(\lambda(A))}{\partial C_{IJ}} = \frac{\partial^{\text{vol}} W(\lambda(A))}{\partial C_{IJ}} + \frac{\partial^{\text{iso}} W(\lambda(A))}{\partial C_{IJ}} \]  

\[ = \frac{1}{2} \frac{\partial^{\text{vol}} W(J)}{\partial J} J (C^{-1})_{IJ} + \frac{1}{2} w_A (M_{IJ}^{(A)})_A \]  

(106.100)

It is obvious that the only material dependent parts are derivatives in the form \( \partial^{\text{vol}} W / \partial J \) and \( w_A \), while the rest is independent of which hyperelastic material model we choose.
106.3.6 Tangent Stiffness Operator

The free energy function decomposition (106.84) is used together with the appropriate definitions made in section (106.3.1) toward the tangent stiffness operator decomposition

\[ \mathcal{L}_{IJKL} = \text{vol}\mathcal{L}_{IJKL} + \text{iso}\mathcal{L}_{IJKL} = 4 \frac{\partial^2 \text{vol}W}{\partial C_{IJ} \partial C_{KL}} + 4 \frac{\partial^2 \text{iso}W}{\partial C_{IJ} \partial C_{KL}} \quad (106.101) \]

The volumetric part \( \frac{\partial^2 \text{vol}W}{\partial C_{IJ} \partial C_{KL}} \) can be written as:

\[
\frac{\partial^2 \text{vol}W}{\partial C_{IJ} \partial C_{KL}} = \frac{1}{4}
\left( J^2 \frac{\partial^2 \text{vol}W}{\partial J \partial J} + J \frac{\partial \text{vol}W}{\partial J} \right) (C^{-1})_{KL}(C^{-1})_{IJ} + \frac{1}{2} J \frac{\partial \text{vol}W}{\partial J} I_{IJKL}^{(C^{-1})} \quad (106.102)
\]

and the complete derivation is again given in appendix 704.3.

The isochoric part \( \frac{\partial^2 \text{iso}W}{\partial C_{IJ} \partial C_{KL}} \) can be written in the following form:

\[
\frac{\partial^2 \text{iso}W}{\partial C_{IJ} \partial C_{KL}} = \frac{1}{4} Y_{AB} (M^{(B)}_{KL})_B (M^{(A)}_{IJ})_A + \frac{1}{2} w_A (M^{(A)}_{IJKL})_A \quad (106.103)
\]

and the complete derivation is given in the appendix (704.4).

Finally, one can write the volumetric and isochoric parts of the tangent stiffness tensors as:

\[
\text{vol}\mathcal{L}_{IJKL} = J^2 \frac{\partial^2 \text{vol}W(J)}{\partial J \partial J} (C^{-1})_{KL}(C^{-1})_{IJ} + J \frac{\partial \text{vol}W(J)}{\partial J} (C^{-1})_{KL}(C^{-1})_{IJ} + 2 J \frac{\partial \text{vol}W(J)}{\partial J} I_{IJKL}^{(C^{-1})} \quad (106.104)
\]

\[
\text{iso}\mathcal{L}_{IJKL} = Y_{AB} (M^{(B)}_{KL})_B (M^{(A)}_{IJ})_A + 2 w_A (M^{(A)}_{IJKL})_A \quad (106.105)
\]

In a similar manner to the stress definitions it is clear that the only material model dependent parts are \( Y_{AB} \) and \( w_A \). The remaining second and fourth order tensors \( M^{(A)}_{IJ} \) and \( M^{(A)}_{IJKL} \) are independent of the choice of the material model. This observation has a practical consequence in that it is possible to create a template derivations for various hyperelastic isotropic material models. Only first and second derivatives of strain energy function with respect to isochoric principal stretches (\( \tilde{\lambda}_A \)) and Jacobian (\( J \)) are needed in addition to the independent tensors, for the determination of various stress and tangent stiffness tensors.
106.3.7 Isotropic Hyperelastic Models

The strain energy function for isotropic solid in terms of principal stretches is represented as:

\[ W = W(\lambda_1, \lambda_2, \lambda_3) \]  

(106.106)

The only restriction is that \( W \) is a symmetric function of \( \lambda_1, \lambda_2, \lambda_3 \), although an appropriate natural configuration condition requires that:

\[ W(1, 1, 1) = 0 \quad \text{and} \quad \frac{\partial W(1, 1, 1)}{\partial \lambda_i} = 0 \]  

(106.107)

The strain energy function \( W \) can either be regarded as a function of principal stretches or the principal invariants of stretches\(^{17} \):

\[ I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \]
\[ I_2 = \lambda_2^2 \lambda_3^2 + \lambda_3^2 \lambda_1^2 + \lambda_1^2 \lambda_2^2 \]
\[ I_3 = \lambda_1^2 \lambda_2^2 \lambda_3^2 \]  

(106.108)

A slightly more general formulation is obtained by using principal stretches in the strain energy function definition. A widely exploited family of compressible hyperelastic models\(^{18} \) are defined (Ogden, 1984) as an infinite series in powers of \((I_1 - 3), (I_2 - 3)\) and \((I_3 - 1)\) as:

\[ W = \sum_{p,q,r=0}^{N \to \infty} c_{pqr} (I_1 - 3)^p (I_2 - 3)^q (I_3 - 1)^r \]  

(106.109)

The regularity condition that \( W \) is continuously differentiable an infinitely number of times is satisfied. The requirement that energy vanishes in the reference configuration is met provided \( c_{000} = 0 \). Reference configuration is stress free iff \( c_{100} + 2c_{010} + c_{001} = 0 \). Isochoric deviatoric decoupling is possible by setting \( c_{pqr} = 0 \ (r = 1, 2, 3, \ldots) \) and \( c_{pqr} = 0 \ (p, q = 1, 2, 3, \ldots) \) to obtain:

\[ W = W^{iso} + W^{vol} \]  

(106.110)

where:

\[ W^{iso} = \sum_{p,q=0}^{N \to \infty} c_{pq0} (I_1 - 3)^p (I_2 - 3)^q \]
\[ W^{vol} = \sum_{r=0}^{N \to \infty} c_{00r} (I_3 - 1)^r \]  

(106.111)

In what follows, we will present a number of widely used strain energy functions for isotropic elastic solids.

\(^{17}\)See also equation (106.61).

\(^{18}\)Used mainly for rubber–like materials.
106.3.7.1 Ogden Model

A very general set of hyperelastic models was defined by Ogden (1984). The strain energy is expressed as a function of principal stretches as:

\[ W = \sum_{r=1}^{\infty} \frac{c_r}{\mu_r} (\lambda_1^{\mu_r} + \lambda_2^{\mu_r} + \lambda_3^{\mu_r} - 3) \]  \hspace{1cm} (106.112)

The isochoric strain energy function can be written as:

\[ \text{iso} W = \sum_{r=1}^{\infty} \frac{c_r}{\mu_r} \left( \tilde{\lambda}_1^{\mu_r} + \tilde{\lambda}_2^{\mu_r} + \tilde{\lambda}_3^{\mu_r} - 1 \right) \]  \hspace{1cm} (106.113)

where the following was used \( \tilde{\lambda}_i = J^{-\frac{1}{3}} \lambda_i \).

Derivatives needed for building tensors \( w_A \) and \( Y_{AB} \) are given by the following formulae:

\[ \frac{\partial \text{iso} W}{\partial \tilde{\lambda}_A} = \sum_{r=1}^{\infty} \frac{c_r}{\mu_r} \left( \tilde{\lambda}_A \right)^{\mu_r - 1} \]  \hspace{1cm} (106.114)

\[ \frac{\partial^2 \text{iso} W}{\partial \tilde{\lambda}_A^2} = \sum_{r=1}^{\infty} c_r (\mu_r - 1) \left( \tilde{\lambda}_A \right)^{\mu_r - 2} \]  \hspace{1cm} (106.115)

\[ \frac{\partial^2 \text{iso} W}{\partial \tilde{\lambda}_A \partial \tilde{\lambda}_B} = 0 \]  \hspace{1cm} (106.116)

106.3.7.2 Neo–Hookean Model

The general isotropic hyperelastic model defined in terms of invariants of principal stretches contains the Neo–Hookean model as special cases. The isochoric part of Neo–Hookean isotropic elastic model can be obtained by selecting \( N = 1, q = 0, c_{p00} = G/2 \), to get:

\[ \text{iso} W = \frac{G}{2} \left( \tilde{\lambda}_1^2 + \tilde{\lambda}_2^2 + \tilde{\lambda}_3^2 - 3 \right) \]  \hspace{1cm} (106.117)

while the volumetric part can be defined by choosing \( N = 2, c_{001} = 0, c_{002} = K_b/2 \), as:

\[ \text{vol} W = \frac{K_b}{2} \left( \lambda_1^2 \lambda_2^2 \lambda_3^2 - 1 \right)^2 = \frac{K_b}{2} (J^2 - 1)^2 \]  \hspace{1cm} (106.118)

where \( G \) and \( K_b \) are the shear and bulk moduli respectively.

Derivatives needed for building tensors \( w_A \) and \( Y_{AB} \) are given by the following formulae:

\[ \frac{\partial \text{iso} W}{\partial \tilde{\lambda}_A} = G \tilde{\lambda}_A \]  \hspace{1cm} (106.119)

\[ \frac{\partial^2 \text{iso} W}{\partial \tilde{\lambda}_A^2} = G \]  \hspace{1cm} (106.120)

\[ \frac{\partial^2 \text{iso} W}{\partial \tilde{\lambda}_A \partial \tilde{\lambda}_B} = 0 \]  \hspace{1cm} (106.121)
106.3.7.3 Mooney–Rivlin Model

Mooney proposed a strain energy function for isochoric behavior of the form:

\[ W^{iso} = \sum_{n=0}^{N \to \infty} a_n \left( \lambda_1^{2n} + \lambda_2^{2n} + \lambda_3^{2n} \right) + a_n \left( \lambda_1^{-2n} + \lambda_2^{-2n} + \lambda_3^{-2n} \right) \]

or

\[ W^{iso} = \sum_{n=0}^{N \to \infty} a_n \left( \bar{\lambda}_1^{2n} + \bar{\lambda}_2^{2n} + \bar{\lambda}_3^{2n} \right) + a_n \left( \bar{\lambda}_1^{-2n} + \bar{\lambda}_2^{-2n} + \bar{\lambda}_3^{-2n} \right) \]

with \( a_n \) and \( b_n \) being the material parameters and a volume preserving constrain \( \lambda_a = 1/(\lambda_b \lambda_c) \), and \( a, b, c \) are cyclic permutations of \((1, 2, 3)\). A more general form was proposed by Rivlin:

\[ W^{iso} = \sum_{p,q=0}^{N \to \infty} c_{pq} (I_1 - 3)^p (I_2 - 3)^q \]

which is actually quite similar to the isochoric part of the general isotropic representation from equation (106.111). Both Mooney and Rivlin strain energy functions become similar, if one chooses to set \( N = 1 \) and \( c_{10} = C_1 \) and \( c_{01} = C_2 \) to obtain:

\[ W^{iso} = \left( C_1 \left( \bar{\lambda}_1 + \bar{\lambda}_2 + \bar{\lambda}_3 - 3 \right) + C_2 \left( \bar{\lambda}_1^{-2} + \bar{\lambda}_2^{-2} + \bar{\lambda}_3^{-2} - 3 \right) \right) \]

\[ = \left( C_1 \left( \bar{I}_1 - 3 \right) + C_2 \left( \bar{I}_2 - 3 \right) \right) \]

Derivatives needed for building tensors \( w_A \) and \( Y_{AB} \) are given by the following formulae:

\[ \frac{\partial W^{iso}}{\partial \lambda_A} = 2 C_1 \bar{\lambda}_A - 2 C_2 \bar{\lambda}_A^{-3} \]

\[ \frac{\partial^2 W^{iso}}{\partial \lambda_A^2} = 2 C_1 + 6 C_2 \bar{\lambda}_A^{-4} \]

\[ \frac{\partial^2 W^{iso}}{\partial \lambda_A \partial \lambda_B} = 0 \]

106.3.7.4 Logarithmic Model

By choosing an alternative set of isochoric principal stretch invariants in the form:

\[ I_1^{ln} = 2 \left( \ln \bar{\lambda}_1 \right)^2 + 2 \left( \ln \bar{\lambda}_2 \right)^2 + 2 \left( \ln \bar{\lambda}_3 \right)^2 \]

\[ = \left( \bar{\lambda}_1^{ln} \right)^2 + \left( \bar{\lambda}_2^{ln} \right)^2 + \left( \bar{\lambda}_3^{ln} \right)^2 \]

\[ I_2^{ln} = 4 \left( \ln \bar{\lambda}_1 \right)^2 \left( \ln \bar{\lambda}_3 \right)^2 + 4 \left( \ln \bar{\lambda}_3 \right)^2 \left( \ln \bar{\lambda}_1 \right)^2 + 4 \left( \ln \bar{\lambda}_1 \right)^2 \left( \ln \bar{\lambda}_2 \right)^2 \]

\[ = \left( \bar{\lambda}_1^{ln} \right)^2 \left( \bar{\lambda}_3^{ln} \right)^2 + \left( \bar{\lambda}_3^{ln} \right)^2 \left( \bar{\lambda}_1^{ln} \right)^2 + \left( \bar{\lambda}_1^{ln} \right)^2 \left( \bar{\lambda}_2^{ln} \right)^2 \]
where the isochoric logarithmic stretch \( \tilde{\lambda}_i^{\ln} \) was used:

\[
\tilde{\lambda}_i^{\ln} = \sqrt{2} \ln \tilde{\lambda}_i = \frac{1}{\sqrt{2}} \ln \tilde{\lambda}_i^2
\]  
(106.129)

The general representation of the isochoric part of the strain energy function in terms of \( \tilde{I}_1^{\ln} \) and \( \tilde{I}_2^{\ln} \) was proposed by Simo and Miehe (1992). A somewhat simpler isochoric strain energy function can be presented in the form:

\[
isoW = G \left( (\ln \tilde{\lambda}_1)^2 + (\ln \tilde{\lambda}_2)^2 + (\ln \tilde{\lambda}_3)^2 \right)
\]  
(106.130)

while the volumetric part is suggested in the form:

\[
volW = \frac{K_b}{2} (\ln J)^2
\]  
(106.131)

Derivatives needed for building tensors \( w_A \) and \( Y_{AB} \) are given by the following formulae:

\[
\frac{\partial isoW}{\partial \tilde{\lambda}_A} = 2 G \left( \tilde{\lambda}_A \right)^{-1}
\]  
(106.132)

\[
\frac{\partial^2 isoW}{\partial \tilde{\lambda}_A^2} = -2 G \left( \tilde{\lambda}_A \right)^{-2}
\]  
(106.133)

\[
\frac{\partial^2 isoW}{\partial \tilde{\lambda}_A \partial \tilde{\lambda}_B} = 0
\]  
(106.134)

\[
\frac{d (volW)}{dJ} = K_b \frac{1}{J} \ln J
\]  
(106.135)

\[
\frac{d^2 (volW)}{dJ^2} = K_b J^{-2} - K_b J^{-2} \ln J
\]  
(106.136)

### 106.3.7.5 Simo–Pister Model

Another form or a volumetric part of strain energy function was proposed by Simo and Pister (1984) in the form:

\[
W_{vol}(J) = \frac{1}{4} K_b \left( J^2 - 1 - 2 \ln J \right)
\]  
(106.137)

The first and second derivatives with respect to \( J \) are then given as:

\[
\frac{d^{volW}(J)}{dJ} = \left( \frac{-2}{4} + 2 \frac{J}{4} \right) K_b
\]  
(106.138)

\[
\frac{d^2^{volW}(J)}{dJ^2} = \left( \frac{2 + \frac{2}{4}}{4} \right) K_b
\]  
(106.139)
106.4 Finite Deformation Hyperelasto–Plasticity

106.4.1 Introduction

The mathematical structure and numerical analysis of classical small deformation elasto–plasticity is generally well established. However, development of large deformation elastic–plastic algorithms for isotropic and anisotropic material models is still a research area. Here, we present a new integration algorithm, based on the multiplicative decomposition of the deformation gradient into elastic and plastic parts. The algorithm is novel in that it is designed to be used with isotropic as well as anisotropic material models. Consistent derivation is based on the idea from the infinitesimal strain algorithm developed earlier by Jeremić and Sture (1997). The algorithm is not an extension of earlier developments, but rather a novel development which consistently utilizes Newton’s method for numerical solution scheme for integrating pertinent constitutive equations. It is also shown that in the limit, the proposed algorithm reduces to the small strain counterpart.

In what follows, we briefly introduce the multiplicative decomposition of the deformation gradient and pertinent constitutive relations. We then proceed to present the numerical algorithm and the algorithmic tangent stiffness tensor consistent with the presented algorithm.

106.4.2 Kinematics

An appropriate generalization of the additive strain decomposition is the multiplicative decomposition of displacement gradient. The motivation for the multiplicative decomposition can be traced back to the early works of Bilby et al. (1957), and Kröner (1960) on micromechanics of crystal dislocations and application to continuum modeling. In the context of large deformation elastoplastic computations, the work by Lee and Liu (1967), Fox (1968) and Lee (1969) stirred an early interest in multiplicative decomposition.

The appropriateness of multiplicative decomposition technique for soils may be justified from the particulate nature of the material. From the micromechanical point of view, plastic deformation in soils arises from slipping, crushing, yielding and plastic bending\(^{19}\) of granules comprising the assembly\(^{20}\). It can certainly be argued that deformations in soils are predominantly plastic, however, reversible deformations could develop from the elasticity of soil grains, and could be relatively large when particles are locked in high density specimens.

The reasoning behind multiplicative decomposition is a rather simple one. If an infinitesimal neighborhood of a body \(x_i, x_i + dx_i\) in an inelastically deformed body is cut–out and unloaded to an unstressed

\(^{19}\)For plate like clay particles.

\(^{20}\)See also Lambe and Whitman (1979) and Sture (1993).
configuration, it would be mapped into \( \hat{x}_i, \hat{x}_i + \hat{d}x_i \). The transformation would be comprised of a rigid body displacement\(^{21}\) and purely elastic unloading. The elastic unloading is a fictitious one, since materials with a strong Baushinger’s effect, unloading will lead to loading in an other stress direction, and, if there are residual stresses, the body must be cut–out in small pieces and then every piece relieved of stresses. The unstressed configuration is thus incompatible and discontinuous. The position \( \hat{x}_i \) is arbitrary, and we may assume a linear relationship between \( dx_i \) and \( \hat{d}x_i \), in the form\(^{22}\):

\[
d\hat{x}_k = (F_{ik}^e)^{-1} \, dx_i
\]

\(^{21}\) Translation and rotation.
\(^{22}\) Referred to same Cartesian coordinate system.
where \((F_{ik}^e)^{-1}\) is not to be understood as a deformation gradient, since it may represent the incompatible, discontinuous deformation of a body. By considering the reference configuration of a body \(dX_i\), then the connection to the current configuration is\(^{23}\):

\[
dx_k = F_{ki}dX_i \Rightarrow d\hat{x}_k = (F_{ik}^e)^{-1} F_{ij}dX_j
\]

so that one can define:

\[
F_{kj}^p \overset{def}{=} (F_{ik}^e)^{-1} F_{ij} \Rightarrow F_{ij}^e \overset{def}{=} F_{ki}^e F_{kj}^p
\]

The plastic part of the deformation gradient, \(F_{kj}^p\) represents micro–mechanically, the irreversible process of slipping, crushing dislocation and macroscopically the irreversible plastic deformation of a body. The elastic part, \(F_{ki}^e\) represents micro–mechanically a pure elastic reversal of deformation for the particulate assembly, macroscopically a linear elastic unloading toward a stress free state of the body, not necessarily a compatible, continuous deformation but rather a fictitious elastic unloading of small cut outs of a deformed particulate assembly or continuum body.

### 106.4.3 Constitutive Relations

We propose the free energy density \(W\), which is defined in \(\bar{\Omega}\), as follows

\[
\rho_0 W(\bar{C}_e^{ij}, \kappa_\alpha) = \rho_0 W^e(\bar{C}_e^{ij}) + \rho_0 W^p(\kappa_\alpha)
\]

where \(W^e(\bar{C}_e^{ij})\) represents a suitable hyperelastic model in terms of the elastic right deformation tensor \(\bar{C}_e^{ij}\), whereas \(W^p(\kappa_\alpha)\) represents the hardening. It has been shown elsewhere (Runesson, 1996), that the pertinent dissipation inequality becomes

\[
D = \bar{T}_{ij} \bar{L}_p^{ij} + \sum_\alpha \bar{K}_\alpha \kappa_\alpha \geq 0
\]

where \(\bar{T}_{ij}\) is the Mandel stress\(^{24}\) and \(\bar{L}_p^{ij}\) is the plastic velocity gradient defined on \(\bar{\Omega}\).

We now define elastic domain \(B\) as

\[
B = \{\bar{T}_{ij}, \bar{K}_\alpha | \Phi(\bar{T}_{ij}, \bar{K}_\alpha) \leq 0\}
\]

When \(\Phi\) is isotropic in \(\bar{T}_{ij}\) (which is the case here) in conjunction with elastic isotropy, we can conclude that \(\bar{T}_{ij}\) is symmetrical and we may replace \(\bar{T}_{ij}\) by \(\tau_{ij}\) in \(\Phi\).

\(^{23}\)See section 106.2.2.

\(^{24}\)See section 106.3.1.
As to the choice of elastic law, it is emphasized that this is largely a matter of convenience since we shall be dealing with small elastic deformations. Here, the Neo–Hookean elastic law is adopted. The generic situation is \( \bar{T}_{ij} = \bar{T}_{ij}(\tilde{U}_{e}^{c}, \bar{J}_{e}) \), where we have used the isochoric/volumetric split of the elastic right stretch tensor as \( \tilde{U}_{e}^{c} = \tilde{U}_{e}^{c}(\bar{J}_{e})^{1/3} \).

The constitutive relations can now be written as

\[
\bar{T}_{ij} = \bar{T}_{ij}(\tilde{U}_{e}^{c}, \bar{J}_{e}) \tag{106.146}
\]

\[
\bar{L}_{p} = \dot{\bar{F}}_{p}^{i} \left( \bar{F}_{p}^{j} \right) = \bar{M}_{ij} \tag{106.147}
\]

\[
\bar{K}_{\alpha} = \bar{K}_{\alpha}(\bar{\kappa}_{\beta}) \tag{106.148}
\]

\[
\dot{\bar{\kappa}}_{\beta} = \mu \frac{\partial \Phi^{*}}{\partial \bar{K}_{\beta}} , \quad \kappa_{\beta}(0) = 0 \tag{106.149}
\]

where \( \bar{L}_{p} = (\bar{F}_{p}^{c})^{-1}F_{lk} \) is the plastic part of the deformation gradient.

### 106.4.4 Implicit Integration Algorithm

The incremental deformation and plastic flow are governed by the system of evolution equations (106.147) and (106.149):

\[
\dot{\bar{F}}_{p}^{i} \left( \bar{F}_{p}^{j} \right) = \mu \bar{M}_{ij} \tag{106.150}
\]

\[
\dot{\bar{\kappa}}_{\beta} = \mu \frac{\partial \Phi^{*}}{\partial \bar{K}_{\beta}} , \quad \kappa_{\beta}(0) = 0 \tag{106.151}
\]

The flow rule from equation (106.150) can be integrated to give

\[
n+1 F_{ij}^{p} = \exp \left( \Delta \mu^{n+1} \bar{M}_{ik} \right) n F_{kj}^{p} \tag{106.152}
\]

By using the multiplicative decomposition

\[
F_{ij} = \bar{F}_{ik}^{c} F_{kj}^{p} \Rightarrow \bar{F}_{ik}^{c} = F_{ij} \left( F_{kj}^{p} \right)^{-1} \tag{106.153}
\]

and equation (106.152) we obtain

\[
n+1 \bar{F}_{ij}^{c} = n+1 F_{im}^{e} \left( n F_{mk}^{p} \right)^{-1} \exp \left( -\Delta \mu^{n+1} \bar{M}_{kj} \right) = n+1 F_{ik}^{c, tr} \exp \left( -\Delta \mu^{n+1} \bar{M}_{kj} \right) \tag{106.154}
\]

where we used that

\[
n+1 F_{ik}^{c, tr} = n+1 F_{im}^{e} \left( n F_{km}^{p} \right)^{-1} \tag{106.155}
\]
The elastic deformation is then
\[
(n+1)C_{ij}^{e} \overset{\text{def}}{=} (n+1)^{T} F_{im}^{e} T_{n+1} F_{mj}^{e} = \exp\left(-\Delta \mu^{n+1} M_{ij}^{e}\right) \left(\frac{n+1}{F_{km}}\right) T_{n+1} F_{jk}^{e, tr} \exp\left(-\Delta \mu^{n+1} M_{ij}^{e}\right)
\]
(106.156)

By recognizing that the exponent of a tensor can be expanded in Taylor series\(^{25}\)
\[
\exp\left(-\Delta \mu^{n+1} M_{ij}^{e}\right) = \delta_{ij} - \Delta \mu^{n+1} M_{ij}^{e} + \frac{1}{2} \left(\Delta \mu^{n+1} M_{is}^{e} \right) \left(\Delta \mu^{n+1} M_{sj}^{e}\right) + \cdots
\]
(106.157)

and by using the first order expansion in the equation (106.156), we obtain
\[
(n+1)C_{ij}^{e} = (\delta_{ir} - \Delta \mu^{n+1} M_{ir}) n+1 C_{ij}^{e, tr} \left(\delta_{ij} - \Delta \mu^{n+1} M_{ij}^{e}\right)
\]
\[
= \left(n+1 C_{il}^{e, tr} - \Delta \mu^{n+1} M_{ir} n+1 C_{rl}^{e, tr}\right) \left(\delta_{ij} - \Delta \mu^{n+1} M_{ij}^{e}\right)
\]
\[
= n+1 C_{ij}^{e, tr} - \Delta \mu^{n+1} M_{ir} n+1 C_{ij}^{e, tr} - \Delta \mu^{n+1} C_{il}^{e, tr} n+1 M_{ij}^{e} + \Delta \mu^{2} n+1 M_{ir} n+1 C_{rl}^{e, tr} n+1 M_{ij}^{e}
\]
(106.158)

**Remark 106.4.1** The Taylor’s series expansion from equation (106.157) is a proper approximation for the general nonsymmetric tensor \(M_{ij}\). That is, the approximate solution given by equation (106.158) is valid for a general anisotropic solid. This contrasts with the spectral decomposition family of solutions\(^{26}\) which are restricted to isotropic solids.

**Remark 106.4.2** Taylor’s series expansion\(^{27}\) is proper for “small” values of plastic flow tensor \(\Delta \mu^{n+1} M_{ij}\).
This is indeed the case for small increments, when \(\Delta \mu \to 0\) which are required for following the equilibrium path for path–dependent solids.

**Remark 106.4.3** In the limit, when the displacements are sufficiently small, the solution (106.158)

---

\(^{25}\)See for example Pearson (1974).

\(^{26}\)See Simo (1992).

\(^{27}\)It should be called MacLaurin’s series expansion, since expansion is about zero plastic flow state (no incremental plastic deformation).
collapses to

\[
\lim_{F_{ij} \to \delta_{ij}} \delta_{ij} + 2^{n+1} \epsilon_{ij} = \delta_{ij} + 2^{n+1} \epsilon_{ij}^{e.tr} \\
- \Delta \mu^{n+1} \bar{M}_{ir} \left( \delta_{rj} + 2^{n+1} \epsilon_{rj}^{e.tr} \right) \\
- \Delta \mu \left( \delta_{il} + 2^{n+1} \epsilon_{il}^{e.tr} \right) n^{n+1} \bar{M}_{ij} \\
+ \Delta \mu^{2} n^{n+1} \bar{M}_{ir} \left( \delta_{il} + 2^{n+1} \epsilon_{il}^{e.tr} \right) n^{n+1} \bar{M}_{ij} \\
= \delta_{ij} + 2^{n+1} \epsilon_{ij}^{e.tr} \\
- \Delta \mu^{n+1} \bar{M}_{ij} - 2 \Delta \mu^{n+1} \bar{M}_{ir} n^{n+1} \epsilon_{rj}^{e.tr} \\
- \Delta \mu^{n+1} \bar{M}_{il} - 2 \Delta \mu^{n+1} \bar{M}_{il} n^{n+1} \epsilon_{il}^{e.tr} \\
+ \Delta \mu^{2} n^{n+1} \bar{M}_{il} n^{n+1} \bar{M}_{ij} + 2 \Delta \mu^{2} n^{n+1} \bar{M}_{ir} n^{n+1} \epsilon_{rj}^{e.tr} n^{n+1} \bar{M}_{ij} \\
= \delta_{ij} + 2^{n+1} \epsilon_{ij}^{e.tr} - 2 \Delta \mu^{n+1} \bar{M}_{ij} \\
\Rightarrow n^{+1} \epsilon_{ij} = n^{+1} \epsilon_{ij}^{tr} - \Delta \mu^{n+1} \bar{M}_{ij} \tag{106.159}
\]

which is a small deformation elastic predictor–plastic corrector equation in strain space. In working out the small deformation counterpart (106.159) it was used that

\[
\lim_{F_{ij} \to \delta_{ij}} F_{ij} \to \delta_{ij} + 2^{n+1} \epsilon_{ij}^{e.tr} = \delta_{ij} + 2^{n+1} \epsilon_{ij}^{e.tr} \\
- \Delta \mu^{n+1} \bar{M}_{ir} \left( \delta_{rj} + 2^{n+1} \epsilon_{rj}^{e.tr} \right) \\
- \Delta \mu \left( \delta_{il} + 2^{n+1} \epsilon_{il}^{e.tr} \right) n^{n+1} \bar{M}_{ij} \\
+ \Delta \mu^{2} n^{n+1} \bar{M}_{ir} \left( \delta_{il} + 2^{n+1} \epsilon_{il}^{e.tr} \right) n^{n+1} \bar{M}_{ij} \\
= \delta_{ij} + 2^{n+1} \epsilon_{ij}^{e.tr} \\
- \Delta \mu^{n+1} \bar{M}_{ij} - 2 \Delta \mu^{n+1} \bar{M}_{ir} n^{n+1} \epsilon_{rj}^{e.tr} \\
- \Delta \mu^{n+1} \bar{M}_{il} - 2 \Delta \mu^{n+1} \bar{M}_{il} n^{n+1} \epsilon_{il}^{e.tr} \\
+ \Delta \mu^{2} n^{n+1} \bar{M}_{il} n^{n+1} \bar{M}_{ij} + 2 \Delta \mu^{2} n^{n+1} \bar{M}_{ir} n^{n+1} \epsilon_{rj}^{e.tr} n^{n+1} \bar{M}_{ij} \\
= \delta_{ij} + 2^{n+1} \epsilon_{ij}^{e.tr} - 2 \Delta \mu^{n+1} \bar{M}_{ij} \\
\Rightarrow n^{+1} \epsilon_{ij} = n^{+1} \epsilon_{ij}^{tr} - \Delta \mu^{n+1} \bar{M}_{ij} \tag{106.159}
\]

By neglecting the higher order term with \( \Delta \mu^{2} \) in equation (106.158), the solution for the right elastic deformation tensor \( n^{+1} \bar{C}_{ij}^{e} \) can be written as

\[
n^{+1} \bar{C}_{ij}^{e} = n^{+1} \bar{C}_{ij}^{e.tr} - \Delta \mu \left( n^{+1} \bar{M}_{ir} n^{+1} \bar{C}_{rj}^{e.tr} + n^{+1} \bar{C}_{il}^{e.tr} n^{+1} \bar{M}_{ij} \right) \tag{106.161}
\]

The hardening rule (106.151) can be integrated to give

\[
n^{+1} \kappa_{\alpha} = n_{\alpha} + \Delta \mu \left. \frac{\partial \Phi^{*}}{\partial K_{\alpha}} \right|_{n+1} \tag{106.162}
\]

**Remark 106.4.4** It is interesting to note that equation (106.161) resembles the elastic predictor–plastic corrector equation for small deformation elastic–plastic incremental analysis. That resemblance will be used to build an iterative solution algorithm in the next section.

The incremental problem is defined by equations (106.161), (106.162), and the constitutive relations

\[
n^{+1} S_{I,J} = 2 \left. \frac{\partial W}{\partial C_{I,J}} \right|_{n+1} \tag{106.163}
\]
\[ n^{+1}K_\alpha = - \left. \frac{\partial W}{\partial \kappa^\alpha} \right|_{n^{+1}} \] (106.164)

and the Karush–Kuhn–Tucker (KKT) conditions

\[ \Delta \mu < 0 \quad ; \quad n^{+1}\Phi \leq 0 \quad ; \quad \Delta \mu n^{+1}\Phi = 0 \] (106.165)

where

\[ \Phi = \Phi(T_{ij}, K_\alpha) \] (106.166)

Remark 106.4.5 The Mandel stress tensor \( \overline{T}_{ij} \) can be obtained from the second Piola–Kirchhoff stress tensor \( \overline{S}_{kj} \) and the right elastic deformation tensor \( \overline{C}_{ik}^e \) as

\[ \overline{T}_{ij} = \overline{C}_{ik}^e \overline{S}_{kj} \] (106.167)

This set of nonlinear equations will be solved with a Newton type procedure, described in the next section. For a given \( n^{+1}F_{ij} \) or \( n^{+1}\overline{C}_{ij}^{tr} \), the upgraded quantities \( n^{+1}S_{IJ} \) and \( n^{+1}K_\alpha \) can be found, then the appropriate pull–back to \( B_0 \) or push–forward to \( B \) will give \( n^{+1}S_{IJ} \) and \( n^{+1}_r^{+1}F_{ij} \).

The elastic predictor, plastic corrector equation

\[ n^{+1}\overline{C}_{ij}^{e} = n^{+1}\overline{C}_{ij}^{e, tr} - \Delta \mu \left( n^{+1}\overline{M}_{ij} + n^{+1}\overline{C}_{ij}^{e, tr} + n^{+1}\overline{C}_{il}^{e, tr} n^{+1}\overline{M}_{lj} \right) \]

is used as a starting point for a Newton iterative algorithm. In previous equation, we have introduced tensor \( Z_{ij} \) to shorten writing. The trial right elastic deformation tensor is defined as

\[ n^{+1}\overline{C}_{ij}^{e, tr} = \left( n^{+1}\overline{F}_{ij}^{e, tr} \right)^T \left( n^{+1}\overline{F}_{iM}^{e, tr} \right)^{-1} \left( n^{+1}\overline{F}_{rM} \right)^T \left( n^{+1}\overline{F}_{JS} \right)^{-1} \] (106.171)

We introduce a tensor of deformation residuals

\[ R_{ij} = \overline{C}_{ij}^{e, tr} - \left( n^{+1}\overline{C}_{ij}^{e, tr} - \Delta \mu n^{+1}Z_{ij} \right) \] (106.172)

Tensor \( R_{ij} \) represents the difference between the current right elastic deformation tensor and the Backward Euler right elastic deformation tensor. The trial right elastic deformation tensor \( n^{+1}\overline{C}_{ij}^{e, tr} \) is maintained fixed during the iteration process. The first order Taylor series expansion can be applied to the equation (106.172) in order to obtain the iterative change, the new residual \( R_{ij}^{\text{new}} \) from the old \( R_{ij}^{\text{old}} \)

\[ R_{ij}^{\text{new}} = R_{ij}^{\text{old}} + d\overline{C}_{ij}^{e, tr} + d(\Delta \mu) n^{+1}Z_{ij} + \Delta \mu \frac{\partial n^{+1}Z_{ij}}{\partial T_{mn}} dT_{mn} + \Delta \mu \frac{\partial n^{+1}Z_{ij}}{\partial K_\alpha} dK_\alpha \] (106.173)
By using that
\[ \bar{T}_{mn} = \bar{C}_{mk} \bar{S}_{kn} \Rightarrow (\bar{C}_{sk})^{-1} \bar{T}_{sn} = \bar{S}_{kn} \]  
(106.174)
we can write
\[
d\bar{T}_{mn} &= d\bar{C}_{mk} \bar{S}_{kn} + \bar{C}_{mk} d\bar{S}_{kn} \\
&= d\bar{C}_{mk} \bar{S}_{kn} + \frac{1}{2} \bar{C}_{mk} \bar{L}_{knpq} d\bar{C}_{pq} \quad \text{from (106.55)} \\
&= d\bar{C}_{mk} (\bar{C}_{sk})^{-1} \bar{T}_{sn} + \frac{1}{2} \bar{C}_{mk} \bar{L}_{knpq} d\bar{C}_{pq} \quad \text{from (106.174)} 
(106.175)
and the equation (106.173) can be rewritten as
\[
R_{ij}^{\text{new}} = R_{ij}^{\text{old}} + d\bar{C}_{ij} + d(\Delta \mu) (n+1)Z_{ij} + \\
+ \Delta \mu \frac{\partial^{n+1}Z_{ij}}{\partial T_{mn}} d\bar{C}_{mk} (\bar{C}_{sk})^{-1} \bar{T}_{sn} + \frac{1}{2} \Delta \mu \frac{\partial^{n+1}Z_{ij}}{\partial T_{mn}} \bar{C}_{mk} \bar{L}_{knpq} d\bar{C}_{pq} + \\
+ \Delta \mu \frac{\partial^{n+1}Z_{ij}}{\partial K_\alpha} dK_\alpha
\]
\[
= R_{ij}^{\text{old}} + d\bar{C}_{ij} + d(\Delta \mu) (n+1)Z_{ij} + \\
+ \Delta \mu \frac{\partial^{n+1}Z_{ij}}{\partial T_{mn}} d\bar{C}_{mk} (\bar{C}_{sk})^{-1} \bar{T}_{sn} + \\
+ \frac{1}{2} \Delta \mu \frac{\partial^{n+1}Z_{ij}}{\partial T_{mn}} \bar{C}_{mk} \bar{L}_{knpq} d\bar{C}_{pq} + \\
+ \Delta \mu \frac{\partial^{n+1}Z_{ij}}{\partial K_\alpha} dK_\alpha
\]
\[
= R_{ij}^{\text{old}} + d\bar{C}_{ij} + d(\Delta \mu) (n+1)Z_{ij} + \\
+ \Delta \mu \frac{\partial^{n+1}Z_{mn}}{\partial T_{ik}} (\bar{C}_{sj})^{-1} \bar{T}_{sk} d\bar{C}_{ij} + \quad \text{dummy indices rearrangement} \\
+ \frac{1}{2} \Delta \mu \frac{\partial^{n+1}Z_{pq}}{\partial T_{mn}} \bar{C}_{mk} \bar{L}_{knij} d\bar{C}_{ij} + \quad \text{dummy indices rearrangement} \\
+ \Delta \mu \frac{\partial^{n+1}Z_{ij}}{\partial K_\alpha} dK_\alpha
\]  
(106.176)
The goal is to have \( R_{ij}^{\text{new}} = 0 \) so one can write

\[
0 = R_{ij}^{\text{old}} + d\tilde{C}_{ij}^e + d(\Delta \mu)^{n+1} Z_{ij} + \\
+ \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial T_{ik}} (\tilde{C}_{sj}^e)^{-1} \tilde{T}_{sk} d\tilde{C}_{ij}^e + \\
+ \frac{1}{2} \Delta \mu \frac{\partial^{n+1} Z_{pq}}{\partial T_{mn}} \tilde{C}_{mk}^e \tilde{C}_{knij}^e d\tilde{C}_{ij}^e + \\
+ \Delta \mu \frac{\partial^{n+1} Z_{ij}}{\partial K_\alpha} dK_\alpha
\]

\[
= R_{ij}^{\text{old}} + d(\Delta \mu)^{n+1} Z_{ij} + \\
+ \Delta \mu \frac{\partial^{n+1} Z_{ij}}{\partial K_\alpha} dK_\alpha + \\
+ d\tilde{C}_{ij}^e + \\
+ \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial T_{ik}} (\tilde{C}_{sj}^e)^{-1} \tilde{T}_{sk} d\tilde{C}_{ij}^e + \\
+ \frac{1}{2} \Delta \mu \frac{\partial^{n+1} Z_{pq}}{\partial T_{mn}} \tilde{C}_{mk}^e \tilde{C}_{knij}^e d\tilde{C}_{ij}^e
\]

\[
= R_{ij}^{\text{old}} + d(\Delta \mu)^{n+1} Z_{ij} + \Delta \mu \frac{\partial^{n+1} Z_{ij}}{\partial K_\alpha} dK_\alpha + \\
+ (\delta_{im} \delta_{nj} + \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial T_{ik}} (\tilde{C}_{sj}^e)^{-1} \tilde{T}_{sk} + \frac{1}{2} \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial T_{pq}} \tilde{C}_{mk}^e \tilde{C}_{knij}^e ) d\tilde{C}_{ij}^e
\]

(106.177)

Upon introducing notation

\[
T_{mnij} = \delta_{im} \delta_{nj} + \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial T_{ik}} (\tilde{C}_{sj}^e)^{-1} \tilde{T}_{sk} + \frac{1}{2} \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial T_{pq}} \tilde{C}_{mk}^e \tilde{C}_{knij}^e
\]

we can solve (106.177) for \( d\tilde{C}_{ij}^e \)

\[
d\tilde{C}_{ij}^e = (T_{mnij})^{-1} \left( -R_{mn}^{\text{old}} - d(\Delta \mu)^{n+1} Z_{mn} - \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial K_\alpha} dK_\alpha \right)
\]

(106.179)

or, by rearranging indices

\[
d\tilde{C}_{pq}^e = (T_{mnpq})^{-1} \left( -R_{mn}^{\text{old}} - d(\Delta \mu)^{n+1} Z_{mn} - \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial K_\alpha} dK_\alpha \right)
\]

(106.180)

By using that

\[
dK_\alpha = \frac{\partial K_\alpha}{\partial \kappa_\beta} d\kappa_\beta = -d(\Delta \mu) \frac{\partial K_\alpha}{\partial \kappa_\beta} \frac{\partial Q}{\partial K_\beta} = -d(\Delta \mu) H_{\alpha \beta} \frac{\partial Q}{\partial K_\beta}
\]

(106.181)

it follows from (106.180)

\[
d\tilde{C}_{pq}^e = (T_{mnpq})^{-1} \left( -R_{mn}^{\text{old}} - d(\Delta \mu)^{n+1} Z_{mn} + \Delta \mu \frac{\partial^{n+1} Z_{mn}}{\partial K_\alpha} d(\Delta \mu) H_{\alpha \beta} \frac{\partial Q}{\partial K_\beta} \right)
\]

(106.182)
A first order Taylor series expansion of a yield function yields

\[
\text{new } \Phi(T_{ij}, K_\alpha) = \text{old } \Phi(T_{ij}, K_\alpha) + \\
+ \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial T_{mn}} dT_{mn} \\
+ \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial K_\alpha} dK_\alpha \\
= \text{old } \Phi(T_{ij}, K_\alpha) + \\
+ \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial T_{mn}} \left( \tilde{C}_{e}^{e} (\bar{C}_{s}^{e})^{-1} \bar{T}_{sn} + \frac{1}{2} \bar{C}_{mk} \bar{L}_{knjq} d\bar{C}_{pq} \right) \\
+ \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial K_\alpha} dK_\alpha \\
= \text{old } \Phi(T_{ij}, K_\alpha) + \\
+ \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial T_{mn}} \left( \bar{C}_{e}^{e} \bar{L}_{knjq} \right) d\bar{C}_{pq} \\
+ \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial K_\alpha} dK_\alpha \tag{106.183}
\]

By using (106.181), equation (106.183) becomes

\[
\text{new } \Phi(T_{ij}, K_\alpha) = \text{old } \Phi(T_{ij}, K_\alpha) + \\
+ \left( \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial T_{mn}} \right) \left( \bar{C}_{e}^{e} \bar{L}_{knjq} \right) d\bar{C}_{pq} \\
- d(\Delta \mu) \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial K_\alpha} H_{\alpha \beta} \frac{\partial \Phi^{*}}{\partial K_{\beta}} \tag{106.184}
\]

Upon introducing the following notation

\[
F_{pq} = \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial T_{pn}} \left( \bar{C}_{e}^{e} \right) d\bar{C}_{pq} \tag{106.185}
\]

and with the solution for \( d\bar{C}_{pq} \) from (106.182), (106.184) becomes

\[
\text{new } \Phi(T_{ij}, K_\alpha) = \text{old } \Phi(T_{ij}, K_\alpha) + \\
+ F_{pq} \left( (T_{mnq})^{-1} \left( -R_{mn}^{old} d(\Delta \mu)^{n+1} Z_{mn} + d(\Delta \mu) \Delta \mu \frac{\partial (n+1) Z_{mn}}{\partial K_\alpha} H_{\alpha \beta} \frac{\partial \Phi^{*}}{\partial K_{\beta}} \right) \right) \\
- d(\Delta \mu) \frac{\partial \Phi(T_{ij}, K_\alpha)}{\partial K_\alpha} H_{\alpha \beta} \frac{\partial \Phi^{*}}{\partial K_{\beta}} \tag{106.186}
\]
After setting \( \Phi(\bar{T}_{ij}, K_\alpha) = 0 \) we can solve for the incremental inconsistency parameter \( d(\Delta \mu) \)

\[
d(\Delta \mu) = \frac{\Phi(\bar{T}_{ij}, K_\alpha) - \Phi(\bar{T}_{ij}, K_\alpha, \Delta \mu)}{\Phi(\bar{T}_{ij}, K_\alpha, \Delta \mu) - \Phi(\bar{T}_{ij}, K_\alpha)} - \frac{n+1}{n} Z_{mn} - \Delta \mu \frac{n+1}{n} Z_{mn} - \frac{1}{\alpha} R_{mn}^{\alpha \beta} \frac{\partial \Phi^*}{\partial K_{\beta}} + \frac{\partial \Phi}{\partial K_{\alpha}} H_{\alpha \beta} \frac{\partial \Phi^*}{\partial K_{\beta}}
\]

(106.187)

**Remark 106.4.6** In the perfectly plastic case, the increment inconsistency parameter \( d(\Delta \mu) \) is

\[
d(\Delta \mu) = \frac{\Phi(\bar{T}_{ij}, K_\alpha) - \Phi(\bar{T}_{ij}, K_\alpha, \Delta \mu)}{\Phi(\bar{T}_{ij}, K_\alpha, \Delta \mu) - \Phi(\bar{T}_{ij}, K_\alpha)} - \frac{n+1}{n} Z_{mn} - \frac{1}{\alpha} R_{mn}^{\alpha \beta} \frac{\partial \Phi^*}{\partial K_{\beta}} + \frac{\partial \Phi}{\partial K_{\alpha}} H_{\alpha \beta} \frac{\partial \Phi^*}{\partial K_{\beta}}
\]

(106.188)

**Remark 106.4.7** In the limit, for small deformations, isotropic response, the increment inconsistency parameter \( d(\Delta \mu) \) becomes

\[
d(\Delta \mu) = \frac{\Phi(\bar{T}_{ij}, K_\alpha) - \Phi(\bar{T}_{ij}, K_\alpha, \Delta \mu)}{\Phi(\bar{T}_{ij}, K_\alpha, \Delta \mu) - \Phi(\bar{T}_{ij}, K_\alpha)} - \frac{n+1}{n} Z_{mn} - \frac{1}{\alpha} R_{mn}^{\alpha \beta} \frac{\partial \Phi^*}{\partial K_{\beta}} + \frac{\partial \Phi}{\partial K_{\alpha}} H_{\alpha \beta} \frac{\partial \Phi^*}{\partial K_{\beta}}
\]

(106.189)

since in the limit, as deformations are getting small

\[
T_{mn} \rightarrow \delta_{pm} \delta_{nq} + \Delta \mu \frac{\partial m_{mn}}{\partial \sigma_{ij}} E_{ijpq} - \frac{1}{2} \epsilon_{pq} \frac{\partial \Phi}{\partial \sigma_{mn}} E_{mn pq}
\]

(106.190)

Upon noting that residual \( R_{pq} \) is defined in strain space, the increment inconsistency parameter \( d(\Delta \mu) \) compares exactly with it’s small strain counterpart (Jeremić and Sture, 1997).

The procedure described below summarizes the implementation of the return algorithm.

**Trial State** Given the right elastic deformations tensor \( \bar{C}_{pq}^{\alpha \beta} \) and a set of hardening variables \( n K_\alpha \) at a specific quadrature point in a finite element, compute the relative deformation gradient \( n+1 f_{ij} \) for a given displacement increment \( \Delta n+1 u_i \), and the right deformation tensor

\[
n+1 f_{ij} = \delta_{ij} + u_{ij}
\]

(106.191)
\[ n+1 \bar{C}_{e,\text{tr}}^{ij} = (n+1 f_{ir} n F_{e}^{r})^{T} (n+1 f_{kl} n F_{e}^{l}) = (n F_{e}^{r})^{T} (n+1 f_{ir})^{T} (n+1 f_{kl} n F_{e}^{l}) \]  

(106.192)

Compute the trial elastic second Piola–Kirchhoff stress and the trial elastic Mandel stress tensor
\[
\begin{align*}
  n+1 \bar{S}_{e,\text{tr}}^{ij} &= 2 \frac{\partial W}{\partial n+1 \bar{C}_{e,\text{tr}}^{ij}} \\
  n+1 T_{e,\text{tr}}^{ij} &= n+1 \bar{C}_{e,\text{tr}}^{il} n+1 \bar{S}_{e,\text{tr}}^{lj}
\end{align*}
\]  

(106.193)

Evaluate the yield function \( n+1 \Phi^{tr}(\bar{T}_{e,\text{tr}}^{ij}, K_{\alpha}) \). If \( n+1 \Phi^{tr} \leq 0 \) there is no plastic flow in current increment
\[
\begin{align*}
  n+1 \bar{C}_{e}^{ij} &= n+1 \bar{C}_{e,\text{tr}}^{ij} \\
  n+1 K_{\alpha} &= n K_{\alpha} \\
  n+1 T_{ij} &= n T_{e,\text{tr}}^{ij}
\end{align*}
\]  

and exit constitutive integration procedure.

**Return Algorithm**  If yield criteria has been violated \((n+1 \Phi^{tr} > 0)\) proceed to step 1.

**step 1.** \( k^{th} \) iteration. Known variables
\[
\begin{align*}
  n+1 \bar{C}_{e}^{(k)} ; \quad n+1 \kappa_{\alpha}^{(k)} ; \quad n+1 K_{\alpha}^{(k)} ; \quad n+1 T_{ij}^{(k)} ; \quad n+1 \Delta \mu^{(k)}
\end{align*}
\]
evaluate the yield function and the residual
\[
\begin{align*}
  \Phi^{(k)} &= \Phi^{n+1 \Phi^{tr}(\bar{T}_{e,\text{tr}}^{(k)}, K_{\alpha}^{(k)})} \\
  R_{ij}^{(k)} &= n+1 \bar{C}_{e,\text{tr}}^{ij} - \left( n+1 \bar{C}_{e,\text{tr}}^{ij} - n+1 \Delta \mu^{(k)} n+1 Z_{ij}^{(k)} \right)
\end{align*}
\]

**step 2.** Check for convergence, \( \Phi^{(k)} \leq NTOL \) and \( \| R_{ij}^{(k)} \| \leq NTOL \). If convergence criteria is satisfied set
\[
\begin{align*}
  n+1 \bar{C}_{e}^{ij} &= n+1 \bar{C}_{e}^{(k)} \\
  n+1 \kappa_{\alpha} &= n+1 \kappa_{\alpha}^{(k)} \\
  n+1 K_{\alpha} &= n+1 K_{\alpha}^{(k)} \\
  n+1 T_{ij} &= n+1 T_{ij}^{(k)} \\
  n+1 \Delta \mu &= n+1 \Delta \mu^{(k)}
\end{align*}
\]
Exit constitutive integration procedure.

**step 3.**\(^{28}\) If convergence is not achieved, i.e. \( \Phi^{(k)} > NTOL \) or \( \| R_{ij}^{(k)} \| > NTOL \) then compute the elastic stiffness tensor \( L_{ijkl} \)
\[
\begin{align*}
  \bar{L}_{ijkl}^{(k)} &= 4 \frac{\partial^{2} W}{\partial C_{e}^{ij}(k) \partial C_{e}^{kl}(k)} \\
\end{align*}
\]  

(106.195)

\(^{28}\text{From step 3. to step 9. all of the variables are in intermediate } n+1 \text{ configuration. For the sake of brevity we are omitting superscript } n+1.\)
step 4. Compute the incremental inconsistency parameter $d(\Delta \mu^{(k+1)})$

$$d(\Delta \mu^{(k+1)}) = \Phi^{(k)} - \mathcal{F}^{(k)}_{mn} R_{mn}^{(k)}$$

where

$$\tilde{H}_{\alpha}^{(k)} = H_{\alpha \beta}^{(k)} \frac{\partial \Phi^{*,(k)}}{\partial K_{\beta}}$$

$$\mathcal{F}_{mn}^{(k)} = \mathcal{F}_{pq}^{(k)} \left( \mathcal{T}_{mnpq}^{(k)} \right)^{-1}$$

$$\mathcal{T}_{mnpq}^{(k)} = \delta_{im} \delta_{nj} + \Delta \mu^{(k)} \frac{\partial \bar{Z}_{mn}^{(k)}}{\partial T_{ik}^{(k)}} \left( \bar{C}_{sj}^{e,(k)} \right)^{-1} \bar{T}_{sk}^{(k)} + \frac{1}{2} \Delta \mu^{(k)} \frac{\partial \bar{Z}_{mn}^{(k)}}{\partial T_{pq}^{(k)}} \bar{C}_{pk}^{e,(k)} \bar{L}_{kj}^{e,(k)}$$

step 5. Updated the inconsistency parameter $\Delta \mu^{(k+1)}$

$$\Delta \mu^{(k+1)} = \Delta \mu^{(k)} + d(\Delta \mu^{(k+1)})$$

step 6. Updated the right deformation tensor, the hardening variable and the Mandel stress

$$d \bar{C}_{pq}^{e,(k+1)} = \left( \mathcal{T}_{mnpq}^{(k)} \right)^{-1} \left( -R_{mn}^{(k)} - d(\Delta \mu^{(k+1)}) n^{+1} \bar{Z}_{mn}^{(k)} + \Delta \mu^{(k)} \frac{\partial \bar{Z}_{mn}^{(k)}}{\partial K_{\alpha}} d(\Delta \mu^{(k+1)}) \tilde{H}_{\alpha}^{(k)} \right)$$

$$d \kappa_{\alpha}^{(k+1)} = d(\Delta \mu^{(k+1)}) \frac{\partial \Phi^{*,(k)}}{\partial K_{\beta}}$$

$$d \bar{T}_{mn}^{(k+1)} = d \bar{C}_{mk}^{e,(k+1)} \left( \bar{C}_{sk}^{e,(k)} \right)^{-1} \bar{T}_{sn}^{(k)} + \frac{1}{2} \bar{C}_{mk}^{e,(k)} \bar{L}_{knpq}^{e,(k)} d \bar{C}_{pq}^{e,(k+1)}$$

step 7. Update right deformation tensor $\bar{C}_{pq}^{e,(k+1)}$, hardening variable $K_{\alpha}^{(k+1)}$ and Mandel stress $\bar{T}_{mn}^{(k+1)}$

$$\bar{C}_{pq}^{e,(k+1)} = \bar{C}_{pq}^{e,(k)} + d(\bar{C}_{pq}^{e,(k+1)})$$

$$\kappa_{\alpha}^{(k+1)} = \kappa_{\alpha}^{(k)} + d(\kappa_{\alpha}^{(k+1)})$$

$$K_{\alpha}^{(k+1)} = K_{\alpha}^{(k)} + d(K_{\alpha}^{(k+1)})$$

$$\bar{T}_{mn}^{(k+1)} = \bar{T}_{mn}^{(k)} + d(\bar{T}_{mn}^{(k+1)})$$

(106.196)
step 8. evaluate the yield function and the residual

\[ \Phi^{(k+1)} = \Phi(\tilde{T}^{e,(k+1)}_{ij}, K^{(k+1)}_{\alpha}) ; \quad R^{(k+1)}_{ij} = \tilde{C}^{e,(k+1)}_{ij} - (\tilde{C}^{e, tr}_{ij} - \Delta \mu^{(k+1)} Z_{ij}) \] (106.203)

step 9. Set \( k = k + 1 \)

\[ \Delta \mu^{(k)} = \Delta \mu^{(k+1)} \]
\[ \tilde{C}^{e,(k)}_{pq} = \tilde{C}^{e,(k+1)}_{pq} \]
\[ \kappa^{(k)}_{\alpha} = \kappa^{(k+1)}_{\alpha} \]
\[ K^{(k)}_{\alpha} = K^{(k+1)}_{\alpha} \]
\[ \tilde{T}^{(k)}_{mn} = \tilde{T}^{(k+1)}_{mn} \] (106.204)

and return to step 2.

106.4.5 Algorithmic Tangent Stiffness Tensor

Starting from the elastic predictor–plastic corrector equation

\[ n+1 \tilde{C}^{e}_{ij} = n+1 \tilde{C}^{e, tr}_{ij} - \Delta \mu^{n+1} Z_{ij} \] (106.205)

and taking the first order Taylor series expansion we obtain

\[ d \tilde{C}^{e}_{ij} = d \tilde{C}^{e, tr}_{ij} - d(\Delta \mu) Z_{ij} - \Delta \mu \frac{\partial Z_{ij}}{\partial T_{mn}} dT_{mn} - \Delta \mu \frac{\partial Z_{ij}}{\partial K_{\alpha}} dK_{\alpha} \]

\[ = d \tilde{C}^{e, tr}_{ij} - d(\Delta \mu) Z_{ij} \]
\[ -\Delta \mu \frac{\partial Z_{ij}}{\partial T_{mn}} \left( d \tilde{C}^{e}_{mk} (\tilde{C}^{e}_{sk})^{-1} \tilde{T}_{sn} + \frac{1}{2} \tilde{C}^{e}_{mk} \tilde{L}_{knpq} d\tilde{C}^{e}_{pq} \right) \] from (106.175)
\[ -\Delta \mu \frac{\partial Z_{ij}}{\partial K_{\alpha}} dK_{\alpha} \] (106.206)

Previous equation can be written as

\[ d \tilde{C}^{e}_{ij} + \Delta \mu \frac{\partial Z_{ij}}{\partial T_{mn}} (\tilde{C}^{e}_{sk})^{-1} \tilde{T}_{sn} d\tilde{C}^{e}_{mk} + \Delta \mu d(\Delta \mu) \frac{\partial Z_{ij}}{\partial T_{mn}} \frac{1}{2} \tilde{C}^{e}_{mk} \tilde{L}_{knpq} d\tilde{C}^{e}_{pq} \]
\[ = d \tilde{C}^{e, tr}_{ij} - d(\Delta \mu) Z_{ij} + \Delta \mu d(\Delta \mu) \frac{\partial Z_{ij}}{\partial K_{\alpha}} H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} \] (106.207)

or as

\[ d \tilde{C}^{e}_{ij} (T_{mnij}) = d \tilde{C}^{e, tr}_{ij} - d(\Delta \mu) Z_{ij} + \Delta \mu d(\Delta \mu) \frac{\partial Z_{ij}}{\partial K_{\alpha}} H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} \] (106.208)

where

\[ T_{mnij} = \delta_{im} \delta_{nj} + \Delta \mu^{(k)} \frac{\partial Z_{mn}^{(k)}}{\partial T_{ik}^{(k)}} (\tilde{C}^{e,(k)}_{sj})^{-1} \tilde{T}_{sk}^{(k)} + \frac{1}{2} \Delta \mu^{(k)} \frac{\partial Z_{mn}^{(k)}}{\partial T_{pq}^{(k)}} \tilde{C}^{e,(k)}_{pk} \tilde{L}_{kqi}^{(k)} \]
The solution for the increment in right elastic deformation tensor is then
\[
d\tilde{C}_{ij} = (T_{mnij})^{-1} \left( d\tilde{C}_{ij}^{e,tr} - d(\Delta \mu) Z_{ij} + \Delta \mu \frac{\partial Z_{ij}}{\partial K} H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} \right)
\]  
(106.209)

We next use the first order Taylor series expansion of yield function \(d\Phi(T_{ij}, K_{\alpha}) = 0\)
\[
\frac{\partial \Phi}{\partial T_{mn}} dT_{mn} + \frac{\partial \Phi}{\partial K_{\alpha}} dK_{\alpha} =
\frac{\partial \Phi}{\partial T_{mn}} \left( d\tilde{C}_{mn}^{e} \left( \tilde{C}_{sk}^{e} \right)^{-1} \tilde{T}_{sn} + \frac{1}{2} \tilde{C}_{mk}^{e} \tilde{L}_{knpq}^{e} d\tilde{C}_{pq}^{e} \right) + \frac{\partial \Phi}{\partial K_{\alpha}} dK_{\alpha} =
\frac{\partial \Phi}{\partial T_{mn}} \left( d\tilde{C}_{mn}^{e} \left( \tilde{C}_{sk}^{e} \right)^{-1} \tilde{T}_{sn} d\tilde{C}_{pq}^{e} + \frac{1}{2} \frac{\partial \Phi}{\partial T_{mn}} \tilde{C}_{mk}^{e} \tilde{L}_{knpq}^{e} d\tilde{C}_{pq}^{e} + \frac{\partial \Phi}{\partial K_{\alpha}} dK_{\alpha} \right) =
\left( \frac{\partial \Phi}{\partial T_{mn}} \left( \tilde{C}_{sk}^{e} \right)^{-1} \tilde{T}_{sn} + \frac{1}{2} \frac{\partial \Phi}{\partial T_{mn}} \tilde{C}_{mk}^{e} \tilde{L}_{knpq}^{e} \right) d\tilde{C}_{pq}^{e} - \frac{\partial \Phi}{\partial K_{\alpha}} d(\Delta \mu) H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} =
F_{pq} d\tilde{C}_{pq}^{e} - \frac{\partial \Phi}{\partial K_{\alpha}} d(\Delta \mu) H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} = 0
\]  
(106.210)

where
\[
F_{pq} = \frac{\partial \Phi}{\partial T_{mn}} \left( \tilde{C}_{sk}^{e} \right)^{-1} \tilde{T}_{sn} + \frac{1}{2} \frac{\partial \Phi}{\partial T_{mn}} \tilde{C}_{mk}^{e} \tilde{L}_{knpq}^{e}
\]  
(106.211)

By using solution for \(d\tilde{C}_{ij}^{e}\) from 106.209 we can write
\[
F_{pq} (T_{mnpq})^{-1} \left( d\tilde{C}_{mn}^{e,tr} - d(\Delta \mu) Z_{mn} + \Delta \mu \frac{\partial Z_{mn}}{\partial K} H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} \right)
- \frac{\partial \Phi}{\partial K_{\alpha}} d(\Delta \mu) H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} = 0
\]  
(106.212)

We are now in the position to solve for the incremental inconsistency parameter \(d(\Delta \mu)\)
\[
d(\Delta \mu) = F_{pq} (T_{mnpq})^{-1} d\tilde{C}_{mn}^{e,tr}
\]  
(106.213)

where we have used \(\Gamma\) to shorten writing
\[
\Gamma = F_{pq} (T_{mnpq})^{-1} n^{+1} Z_{mn} - \Delta \mu F_{pq} (T_{mnpq})^{-1} \frac{\partial^{n+1} Z_{mn}}{\partial K_{\alpha}} H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} + \frac{\partial \Phi}{\partial K_{\alpha}} H_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}}
\]  
(106.214)

Since
\[
dS_{kn} = \frac{1}{2} \tilde{L}_{knpq}^{e} d\tilde{C}_{pq}^{e}
\]  
(106.215)
and by using 106.209 we can write

\[
\begin{align*}
\frac{dC}{\bar{\tau}}_{pq} = \\
(T_{mn pq})^{-1} \left( \delta_{mv} \delta_{nt} - \frac{F_{op} (T_{rs op})^{-1} \delta_{rv} \delta_{st}}{\Gamma} \right) Z_{mn} + \\
\Delta \mu \frac{F_{op} (T_{rs op})^{-1} \delta_{rv} \delta_{st}}{\Gamma} \frac{\partial Z_{ij}}{\partial K_{\alpha}} H_{\alpha \beta} \frac{\partial \Phi^{*}}{\partial K_{\beta}} \Bigg) d\bar{C}_{tr e} \tag{106.216}
\end{align*}
\]

Then

\[
\begin{align*}
\frac{dC}{\bar{\tau}}_{pq} = \mathcal{P}_{pq vt} d\bar{C}_{e tr} \tag{106.217}
\end{align*}
\]

where

\[
\begin{align*}
\mathcal{P}_{pq vt} = \\
(T_{mn pq})^{-1} \left( \delta_{mv} \delta_{nt} - \frac{F_{op} (T_{rs op})^{-1} \delta_{rv} \delta_{st}}{\Gamma} \right) Z_{mn} + \\
\Delta \mu \frac{F_{op} (T_{rs op})^{-1} \delta_{rv} \delta_{st}}{\Gamma} \frac{\partial Z_{ij}}{\partial K_{\alpha}} H_{\alpha \beta} \frac{\partial \Phi^{*}}{\partial K_{\beta}} \Bigg)
\end{align*}
\]

Algorithmic tangent stiffness tensor \(\bar{\mathcal{L}}_{ijkl}\) (in intermediate configuration \(\bar{\Omega}\)) is then defined as

\[
\begin{align*}
\bar{\mathcal{L}}^{ATS}_{kn vt} = \bar{\mathcal{L}}^{e}_{knpq} \mathcal{P}_{pq vt} \tag{106.219}
\end{align*}
\]

Pull–back to the reference configuration \(\Omega_{0}\) yields the algorithmic tangent stiffness tensor \(\mathcal{L}_{ijkl}\) in reference configuration \(\Omega_{0}\)

\[
\begin{align*}
n+1 \mathcal{L}^{ATS}_{ijkl} = n+1 F_{im}^{p} n+1 F_{jn}^{p} n+1 F_{kr}^{p} n+1 F_{ls}^{p} n+1 \mathcal{L}^{ATS}_{mn rs} \tag{106.220}
\end{align*}
\]

Remark 106.4.8 In the limit, for small deformations, isotropic response, the Algorithmic Tangent Stiffness tensor \(\mathcal{L}^{ATS}_{ijkl}\) becomes

\[
\begin{align*}
\lim \mathcal{L}^{ATS}_{vtpq} = E^{ATS}_{vtpq} = \\
E_{kn pq} \left( \frac{\gamma_{mnpq}^{-1}}{\Gamma} \left( \delta_{mv} \delta_{nt} - \frac{n_{cd} E_{c dab} y_{v tab}^{-1} H_{mn}}{\Gamma} \right) \right) \tag{106.221}
\end{align*}
\]
since

\[
\lim \mathcal{T}_{mnpq} = \mathcal{Y}_{mnpq} = \delta_{pm} \delta_{nq} + \Delta \mu \frac{\partial Z_{mn}}{\partial T_{pk}} (\mathcal{C}_{sq})^{-1} \mathcal{T}_{sk} + \frac{1}{2} \Delta \mu \frac{\partial Z_{mn}}{\partial T_{rs}} \mathcal{C}_{rk} \mathcal{C}_{kspq}
\]

\[
= \delta_{pm} \delta_{nq} + \Delta \mu \frac{\partial m_{mn}}{\partial \sigma_{rs}} \mathcal{E}_{kspq}
\]

(106.222)

\[
\lim \mathcal{F}_{ab} = \lim \left( \frac{\partial \Phi}{\partial T_{ad}} (\mathcal{C}_{sb})^{-1} \mathcal{T}_{sd} + \frac{1}{2} \frac{\partial \Phi}{\partial T_{cd}} \mathcal{C}_{ck} \mathcal{E}_{k dab} \right)
\]

\[
= \frac{1}{2} n_{cd} \mathcal{E}_{cdab}
\]

(106.223)

\[
\mathcal{H}_{mn} = m_{mn} - \Delta \mu \frac{\partial m_{mn}}{\partial K_{\alpha}} \mathcal{H}_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}}
\]

(106.224)

\[
\lim \Gamma = \lim \left( \mathcal{F}_{pq} (\mathcal{T}_{mnpq})^{-1} \partial^{n+1} Z_{mn} - \Delta \mu \mathcal{F}_{pq} (\mathcal{T}_{mnpq})^{-1} \frac{\partial^{n+1} Z_{mn}}{\partial K_{\alpha}} \mathcal{H}_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} + \frac{\partial \Phi^*}{\partial K_{\alpha}} \mathcal{H}_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} \right)
\]

\[
= n_{ab} \mathcal{E}_{abpq} \mathcal{Y}_{mnpq}^{-1} m_{mn} - \Delta \mu n_{ab} \mathcal{E}_{abpq} \mathcal{Y}_{mnpq}^{-1} \frac{\partial m_{mn}}{\partial K_{\alpha}} \mathcal{H}_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}} + \frac{\partial \Phi^*}{\partial K_{\alpha}} \mathcal{H}_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}}
\]

\[
= n_{ab} \mathcal{R}_{abmn} \mathcal{H}_{mn} + \frac{\partial \Phi}{\partial K_{\alpha}} \mathcal{H}_{\alpha\beta} \frac{\partial \Phi^*}{\partial K_{\beta}}
\]

(106.225)

It is noted that the Algorithmic Tangent Stiffness tensor given by 106.221 compares exactly with it's small strain counterpart (Jeremić and Sture, 1997).
Chapter 107

Solution of Static Equilibrium Equations

(In collaboration with Dr. Yuan Feng)
107.1 Chapter Summary and Highlights

107.2 The Residual Force Equations

This chapter is based on Felippa (1993).

In previous Chapters we have derived the basic equations for (material and/or geometric) nonlinear analysis of solids. Discretization of such problems by finite element methods results in a set of nonlinear algebraic equations called residual force equations:

\[ r(u, \lambda) = f_{int}(u) - \lambda f_{ext} = 0 \]  

(107.1)

where \( f_{int}(u) \) are the internal forces which are functions of the displacements, \( u \), the vector \( f_{ext} \) is a fixed external loading vector and the scalar \( \lambda \) is a load-level parameter that multiplies \( f_{ext} \). Equation (107.1) describes the case of proportional loading in which the loading pattern is kept fixed. All solution procedures of practical importance are strongly rooted in the idea of ”advancing the solution” by continuation. Except in very simple problems, the continuation process is multilevel and involves hierarchical breakdown into stages, incremental steps and iterative steps. Processing a complex nonlinear problem generally involves performing a series of analysis stages. Multiple control parameters are not varied independently in each stage and may therefore be characterized by a single stage control parameter \( \lambda \). Stages are only weakly coupled in the sense that end solution of one may provide the starting point for another.

107.3 Constraining the Residual Force Equations

Various forms of path following methods\(^1\) have stemmed from the original work of Riks (1972), Riks (1979) and Wempner (1971). They aimed at finding the intersection of equation (107.1) with \( s = \text{constant} \) where \( s \) is the arc-length, defined as\(^2\):

\[ s = \int ds \]  

(107.2)

where:

\[ ds = \sqrt{\frac{\psi^2}{u_{ref}^2} du + d\lambda^2 \psi^2} \]  

(107.3)

\(^1\) also called arc-length methods with various methods of approximating the exact length of an arc.

\(^2\) A bit different form in that it is scaled with scaling matrix \( S \), introduced by Felippa (1984).
Differential form (107.3) can be replaced with an incremental form:

\[
a = (\Delta s)^2 - (\Delta l)^2 = \left( \frac{\psi_u^2}{u_{ref}} \Delta u^T S \Delta u + \Delta \lambda^2 \psi_f^2 \right) - (\Delta l)^2
\]

(107.4)

where \(\Delta l\) is the radius of the desired intersection\(^3\) and represents an approximation to the incremental arc length. Scaling matrix \(S\) is usually diagonal non-negative matrix that scales the state vector \(\Delta u\) and \(u_{ref}\) is a reference value with the dimension of \(\sqrt{\Delta u^T S \Delta u}\). It is important to note that the vector \(\Delta u\) and scalar \(\Delta \lambda\) are incremental and not iterative values, and are starting from the last converged equilibrium state.

---

\(^3\)See Figure (107.1).

\(^4\)\(n\) unknown displacement variables and on extra unknown in the form of load parameter.

---

Figure 107.1: Spherical arc-length method and notation for one DOF system.

The main essence of the arc-length methods is that the load parameter \(\lambda\) becomes a variable. With load parameter \(\lambda\) variable we are dealing with \(n + 1\) unknowns\(^4\). In order to solve this problem we have \(n\) equilibrium equations (107.1) and the one constraint equation (107.4). We can solve the augmented
system of \( n + 1 \) equations by applying the Newton-Raphson\(^5\) method to equations (107.1) and (107.4)

\[
\mathbf{r}^{\text{new}}(\mathbf{u}, \lambda) = \mathbf{r}^{\text{old}}(\mathbf{u}, \lambda) + \frac{\partial \mathbf{r}(\mathbf{u}, \lambda)}{\partial \mathbf{u}} \delta \mathbf{u} + \frac{\partial \mathbf{r}(\mathbf{u}, \lambda)}{\partial \lambda} \delta \lambda = 0
\]

(107.5)

\[
\mathbf{a}^{\text{new}} = \mathbf{a}^{\text{old}} + 2 \psi^2 \mathbf{u} \mathbf{u}^T \Delta \mathbf{u} \mathbf{S} \delta \mathbf{u} + 2 \Delta \lambda \mathbf{\lambda} \psi_f^2 = 0
\]

(107.6)

where \( \mathbf{K}_t = \frac{\partial \mathbf{r}(\mathbf{u}, \lambda)}{\partial \mathbf{u}} \) is the tangent stiffness matrix. The aim is to have \( \mathbf{r}^{\text{new}}(\mathbf{u}, \lambda) = 0 \) and \( \mathbf{a}^{\text{new}} = 0 \) so the previous system can be written as:

\[
\begin{bmatrix}
\mathbf{K}_t & -\mathbf{f}_{\text{ext}} \\
2 \psi_u^2 \Delta \mathbf{u}^T \mathbf{S} & 2 \Delta \lambda \psi_f^2
\end{bmatrix}
\begin{bmatrix}
\delta \mathbf{u} \\
\delta \lambda
\end{bmatrix}
= -
\begin{bmatrix}
\mathbf{r}^{\text{old}} \\
\mathbf{a}^{\text{old}}
\end{bmatrix}
\]

(107.7)

One can solve previous system of two equations for \( \delta \mathbf{u} \) and \( \delta \lambda \):

\[
\begin{bmatrix}
\delta \mathbf{u} \\
\delta \lambda
\end{bmatrix}
= -
\begin{bmatrix}
\mathbf{K}_t & -\mathbf{f}_{\text{ext}} \\
2 \psi_u^2 \Delta \mathbf{u}^T \mathbf{S} & 2 \Delta \lambda \psi_f^2
\end{bmatrix}^{-1}
\begin{bmatrix}
\mathbf{r}^{\text{old}} \\
\mathbf{a}^{\text{old}}
\end{bmatrix}
\]

(107.8)

or by defining the augmented stiffness matrix\(^6\) \( \mathbf{K} \) as:

\[
\mathbf{K} =
\begin{bmatrix}
\mathbf{K}_t & -\mathbf{f}_{\text{ext}} \\
2 \psi_u^2 \Delta \mathbf{u}^T \mathbf{S} & 2 \Delta \lambda \psi_f^2
\end{bmatrix}
\]

(107.9)

the equation (107.8) can be written as:

\[
\begin{bmatrix}
\delta \mathbf{u} \\
\delta \lambda
\end{bmatrix}
= -\mathbf{K}^{-1}
\begin{bmatrix}
\mathbf{r}^{\text{old}} \\
\mathbf{a}^{\text{old}}
\end{bmatrix}
\]

(107.10)

It should be mentioned that the augmented stiffness matrix remains non-singular even if \( \mathbf{K}_t \) is singular.

---

\(^5\)By using a truncated Taylor series expansion.

\(^6\)Or augmented Jacobian.
107.4 Load Control

107.5 Displacement Control

107.6 Generalized, Hyper-Spherical Arc-Length Control

In section (107.3) we have introduced a constraining equation that is intended to reduce the so called drift error in the incremental nonlinear finite element procedure. The constraining equation was given in a rather general form. Some further comments and observations are in order. By assigning various numbers to parameters $\psi_u$, $\psi_f$, $S$ and $u_{ref}$ one can obtain different constraining schemes from (107.4).

$$\psi_u \psi_f = \psi_u \psi_f < \psi_u \psi_f << \Delta l$$

$$\lambda_f$$

Displacement $u$

Figure 107.2: Influence of $\psi_u$ and $\psi_f$ on the constraint surface shape.

Coefficients $\psi_u$ and $\psi_f$ may not be simultaneously zero. Useful choices for $S$ are $I$, $K_t$ and $\text{diag}(K_t)$. If $S = I$ and $u_{ref} = 1$ the method is called the arclength method\textsuperscript{7}. If we choose $S = \text{diag}(K_t)$ nice scaling is obtained\textsuperscript{8} but otherwise no physical meaning can be attributed to this scaling type. With $S = K_t$ and $\psi_f \equiv 0$ one ends up with something very similar to the external work constraint of Bathe and Dvorkin (1983). A rather general equation (107.4) can be further specialized to load ($\lambda$) control with $\psi_u \equiv 0$; $\psi_f \equiv 1$ and state control\textsuperscript{9} with $\psi_u \equiv 1$; $\psi_f \equiv 0$ and $S = I$. In the finite element literature, the term displacement control has been traditionally associated with the case in which only one of the components of the displacement vector $u$\textsuperscript{10} is specified. This may be regarded either as a variant of state

\textsuperscript{7}It actually reduces to the original work of fRiks (1972), Riks (1979) and Wempner (1971).

\textsuperscript{8}For example if FEM model includes both translational and rotational DOFs.

\textsuperscript{9}That is the cylindrical constraint, or general displacement control.

\textsuperscript{10}Say $u_i$. 
control, in which a norm that singles out the $i$th component is used, or as a variant of the $\lambda$ control if the control parameter is taken as $\lambda u_i$. It is, of course, possible to make the previous parameters variable, functions of different unknowns. For example if one defines $u_{ref} = \Delta u^T S \Delta u$ then close to the limit point $\Delta u \to 0 \Rightarrow \frac{\psi^2}{u_{ref}} \gg \psi^2$ that makes our constraint from equation (107.4) behave like state control. One important aspect of scaling constraint equations by using $S = \text{diag}(K_t)$ or $S = K_t$ is the possibility of non–positive definite stiffness matrix $K_t$. It usually happens that after the limit point is passed, at least one of the eigenvalues of $K_t$ is non–positive, thus rendering the constraint hypersurface non–convex.

In order to get better control of the solution to the system of equations (107.10) one may directly introduce the constraint from equation (107.6) by following the approach proposed by Batoz and Dhatt (1979), as described by Crisfield (1991) and Felippa (1993).

The iterative displacement $\delta u$ is split into two parts, and with the Newton change at the new unknown load level:

$$\lambda^{new} = \lambda^{old} + \delta \lambda$$

(107.11)

becomes:

$$\delta u = -K_t^{-1} r(u^{old}, \lambda) = -K_t^{-1} \left( f_{int}(u^{old}) - \lambda^{new} f_{ext} \right)$$

$$= -K_t^{-1} \left( f_{int}(u^{old}) - \left( \lambda^{old} + \delta \lambda \right) f_{ext} \right) = -K_t^{-1} \left( f_{int}(u^{old}) - \lambda^{old} f_{ext} - \delta \lambda f_{ext} \right)$$

$$= -K_t^{-1} \left( f_{ext}(u^{old}, \lambda^{old}) - \delta \lambda f_{ext} \right) = -K_t^{-1} r^{old} + \delta \lambda K_t^{-1} f_{ext} = \delta \bar{u} + \delta \lambda \delta u_t$$  

(107.12)

where $\delta u_t = K_t^{-1} f_{ext}$ is the displacement vector corresponding to the fixed load vector $f_{ext}$, and $\delta \bar{u}$ is an iterative change that would stem from the standard load-controlled Newton-Raphson, at a fixed load level $\lambda^{old}$. With the solution\(^{11}\) for the $\delta u$ from (107.12), the new incremental displacements are:

$$\Delta u^{new} = \Delta u^{old} + \delta u = \Delta u^{old} + \delta \bar{u} + \delta \lambda \delta u_t$$

(107.13)

where $\delta \lambda$ is the only unknown. The constraint from equation (107.4) can be used here, and by rewriting it as:

\(^{11}\)But having in mind that $\delta \lambda$ is still unknown!
\[
\left( \frac{\psi^2}{u_{ref}^2} (\Delta u_{new}^T) S (\Delta u_{new}) + (\Delta \lambda_{new})^2 \psi_f^2 \right) = (\Delta l)^2
\]  
(107.14)

then by substituting \( \Delta u_{new} \) from equation (107.13) into equation (107.14) and by recalling that \( \lambda_{new} = \lambda_{old} + \delta \lambda \) one ends up with the following quadratic scalar equation:

\[
\left( \frac{\psi^2}{u_{ref}^2} (\Delta u_{old}^T + \delta \vec{u} + \delta \lambda \delta \vec{u}) S (\Delta u_{old}^T + \delta \vec{u} + \delta \lambda \delta \vec{u}) + (\Delta \lambda_{old} + \delta \lambda)^2 \psi_f^2 \right) = (\Delta l)^2
\]  
(107.15)

or, by collecting terms:

\[
\left( \frac{\psi^2}{u_{ref}^2} \delta \vec{u}^T S \delta \vec{u} + \psi_f^2 \right) \delta \lambda^2 + 2 \left( \frac{\psi^2}{u_{ref}^2} \delta \vec{u}^T S (\Delta u_{old}^T + \delta \vec{u}) + \Delta \lambda_{old} \psi_f^2 \right) \delta \lambda + \left( \frac{\psi^2}{u_{ref}^2} (\Delta u_{old}^T + \delta \vec{u})^T S (\Delta u_{old}^T + \delta \vec{u}) - \Delta l^2 + (\Delta \lambda_{old})^2 \psi_f^2 \right) = 0
\]  
(107.16)

or:

\[
a_1 \delta \lambda^2 + 2a_2 \delta \lambda + a_3 = 0
\]  
(107.17)

where:

\[
a_1 = \frac{\psi^2}{u_{ref}^2} \delta \vec{u}^T S \delta \vec{u} + \psi_f^2
\]

\[
a_2 = 2 \left( \frac{\psi^2}{u_{ref}^2} \delta \vec{u}^T S (\Delta u_{old}^T + \delta \vec{u}) + \Delta \lambda_{old} \psi_f^2 \right)
\]

\[
a_3 = \frac{\psi^2}{u_{ref}^2} (\Delta u_{old}^T + \delta \vec{u})^T S (\Delta u_{old}^T + \delta \vec{u}) - \Delta l^2 + (\Delta \lambda_{old})^2 \psi_f^2
\]

The quadratic scalar equation (107.17) can be solved for \( \delta \lambda \):

\[
\delta \lambda = \delta \lambda_1 = \frac{-a_2 + \sqrt{a_2^2 - a_1a_3}}{a_1} \quad ; \quad \delta \lambda = \delta \lambda_2 = \frac{-a_2 - \sqrt{a_2^2 - a_1a_3}}{a_1}
\]  
(107.18)
or, if $a_1 = 0$, then:

$$
\delta \lambda = -\frac{a_3}{2a_2}
$$  \hfill (107.19)

and then the complete change is defined from equation (107.13):

$$
\Delta u^{new} = \Delta u^{old} + \delta \bar{u} + \delta \lambda \delta u_t
$$  \hfill (107.20)

An ambiguity is introduced in the solution for $\delta \lambda$ from (107.18). The tangent at the regular point on the equilibrium path has two possible directions, which generally intersect the constraint hypersurface at two points. However, some exceptions from that rule are possible, so the solutions from (107.18) can be categorized as:

- **Real roots of opposite signs.** This occurs when the iteration process converges normally and there is no limit or turning point enclosed by the constraint hypersurface. The root is chosen by applying one of the schemes proposed below.

- **Real roots of equal sign opposite to that of $\Delta \lambda^{old}$.** This usually happens when going over a "flat" limit point.

- **Real roots of equal sign same as that of $\Delta \lambda^{old}$.** This is an unusual case. It may signal a turning point or be triggered by erratic iteration behavior.

- **Complex roots.** This is an unusual case too. It may signal a sharp turning point, a bifurcation point, erratic or divergent iterates.

For the first two cases, the correct $\Delta \lambda$ can be chosen by applying one of the following schemes.

### 107.6.1 Traversing Equilibrium Path in Positive Sense

#### 107.6.1.1 Positive External Work

The simplest rule requires that the external work expenditure over the predictor step be positive:

$$
\Delta W = f_{ext}^T \Delta u = f_{ext}^T K_t^{-1} f_{ext} \delta \lambda > 0
$$  \hfill (107.21)

The simple conclusion is that $\delta \lambda$ should have the sign of $f_{ext}^T K_t^{-1} f_{ext}$. This condition is particularly effective at limit points. However, it fails when $f_{ext}$ and $K_t^{-1} f_{ext}$ are orthogonal:

$$
f_{ext}^T K_t^{-1} f_{ext} = 0
$$  \hfill (107.22)

This can happen at:
• Bifurcation points,

• Turning points,

Figure 107.3: Simple illustration of Bifurcation and Turning point.

The treatment of bifurcation points is of a rather special nature and is left for the near future. Turning points\textsuperscript{12} can be traversed by a modification of a previous rule, as described in the next section.

107.6.1.2 Angle Criterion

Near a turning point application of the positive work rule (107.21) causes the path to double back upon itself. When it crosses the turning point it reverses so the turning point becomes impassable. Physically, a positive work rule is incorrect because in passing a turning point the structure releases external work until another turning point is encountered.

To pass a turning point imposing a condition on the angle of the prediction vector proves more effective. The idea is to compute both solutions $\delta \lambda_1$ and $\delta \lambda_2$ and then both $\Delta p_1^{\text{new}}$ and $\Delta u_1^{\text{new}}$:

\[
\Delta u_1^{\text{new}} = \Delta u^{\text{old}} + \delta \bar{u} + \delta \lambda_1 \delta u_i
\]  
\[
\Delta u_2^{\text{new}} = \Delta u^{\text{old}} + \delta \bar{u} + \delta \lambda_2 \delta u_i
\]

\textsuperscript{12}One might ask "why treating turning points in a material nonlinear analysis?". The answer is rather simple: "try to prevent all unnecessary surprises". For a good account of some of surprises in material nonlinear analysis one might take a look at some examples in \textit{Crisfield (1991)} pp. 270.
The one that lies closest to the old incremental step direction $\Delta u^{old}$ is the one sought. This should prevent the solution from double backing. The procedure can be implemented by finding the solution with the minimum angle between $\Delta u^{old}$ and $\Delta p^{new}$, and hence the maximum cosine of the angle:

$$\cos \phi = \frac{(\Delta u^{old})^T (\Delta u^{new})}{\|\Delta u^{old}\| \|\Delta u^{new}\|} = \frac{(\Delta u^{old})^T (\Delta u^{old} + \delta \dot{u} + \delta \lambda \delta u_t)}{\|\Delta u^{old}\| \|\Delta u^{old} + \delta \dot{u} + \delta \lambda \delta u_t\|}$$ (107.25)

where $\delta \dot{u} = -K^{-1}r^{old}$ and $\delta u_t = K^{-1}f_{ext}$. Once the turning point has been crossed, the work criterion should be reversed so the external work is negative.

By directly introducing the constraint from equation (107.6) and following the method through equations (107.12) to (107.25) a limitation is introduced. Precisely at the limit point, $K_t$ is singular and the equations cannot be solved. However, Batoz and Dhatt (1979) and Crisfield (1991) report that no such problem has occurred, because one appears never to arrive precisely at limit point.

### 107.6.2 Predictor step

The predictor solution is achieved by applying one forward Euler, explicit step from the last obtained equilibrium point:

$$\Delta u_p = K_t^{-1} \Delta q_e = \Delta \lambda_p K_t^{-1} f_{ext} = \Delta \lambda_p \delta u_t$$ (107.26)

where $K_t$ is the tangent stiffness matrix at the beginning of increment. Substituting equation (107.26) into the constraint equation (107.14) one obtains:

$$\left( \frac{\psi^2}{u^2_{ref}} (\Delta u^{new})^T S (\Delta u^{new}) + (\Delta \lambda^{new})^2 \psi_f^2 \right) = \left( \frac{\psi^2}{u^2_{ref}} \Delta \lambda_p^2 \delta u_t^T S \delta u_t \right) + (\Delta \lambda_p)^2 \psi_f^2 = \Delta \lambda_p^2 \left( \frac{\psi^2}{u^2_{ref}} \delta u_t^T S \delta u_t + \psi_f^2 \right) = (\Delta l)^2$$ (107.27)

The solution for $\Delta \lambda_p$ is readily found:

$$\Delta \lambda_p = \pm \frac{\Delta l}{\sqrt{\frac{\psi^2}{u^2_{ref}} \delta u_t^T S \delta u_t + \psi_f^2}}$$ (107.28)
where $\Delta l > 0$ is the given increment length. The absolute value of $|\delta u^T t S \delta u_t|$ is needed if the stiffness matrix is chosen as a scaling matrix, i.e. $S = K_t$, since, after passing limit point, the stiffness matrix is non-positive definite so $\delta u^T t S \delta u_t \leq 0$. The question of choosing the right sign + or − in (107.28) is still a vigorous research topic. In the simplified procedure\(^{13}\) the negative sign − is chosen with respect to the occurrence of one negative pivot during factorization of the tangent stiffness matrix $K_t$. If more than one pivot happens to be negative, one is advised\(^{14}\) to stop the analysis and try to restart from previously converged solution with smaller step size.

### 107.6.3 Automatic Increments

A number of workers have advocated different strategies for controlling the step length size. In this work we will follow the strategy advocated by Crisfield (1991). The idea is to find the new incremental length by applying:

\[
\Delta l^{\text{new}} = \Delta l^{\text{old}} \left( \frac{I_{\text{desired}}}{I_{\text{old}}} \right)^n \tag{107.29}
\]

where $\Delta l^{\text{old}}$ is the old incremental factor for which $I_{\text{old}}$ iterations were required, $I_{\text{desired}}$ is the input, desired number of iterations\(^{15}\) and the parameter $n$ is set to $\frac{1}{2}$ as suggested by Ramm (1982) Ramm (1981).

### 107.6.4 Convergence Criteria

Introduction of an iterative scheme calls for the introduction of an iteration termination test. There are several convergence criteria that can be applied.

- **Absolute Displacement Convergence Criteria.** The change in the last correction $\delta u$ of the state vector $u$, measured in an appropriate norm, should not exceed a given tolerance $\epsilon_u$. For example, if we use Euclidean norm\(^ {16}\) the termination criteria can be written as:

\[
\|\delta u\|_{\text{absolute}} = \sqrt{(\delta u)^T S (\delta u)} \leq \epsilon_u \tag{107.30}
\]

Scaling matrix $S$ is used in order to ensure that for a problem involving mixed units\(^ {17}\), all parameters have the same unit. Here, an obvious choice for the scaling matrix is $S = \text{diag}(K_t^{-1})$. If, on the

\[^{13}\text{Which is not guaranteed to work if one takes into account bifurcation phenomena.}\]
\[^{14}\text{For more details see Crisfield (1991).}\]
\[^{15}\text{Say } I_{\text{desired}} \approx 3.}\]
\[^{16}\text{The so called 2–norm.}\]
\[^{17}\text{For example, if rotations and displacements are involved.}\]
other hand we don’t have mixed variables in state vector \( \mathbf{u} \) then the simplest choice for scaling matrix is \( S = \mathbf{I} \).

Currently used within F EI and the Real-ESSI program is an absolute convergence criteria with unit scaling matrix \( S = \mathbf{I} \). This means that absolute tolerance criteria mixes units for different Degrees of Freedom (DoFs). Supplied tolerance is converted to basic units of the system (meter and Newton) and that is used for comparison and convergence decisions.

- **Relative Displacement Convergence Criteria.** It is beneficial to use the relative convergence criteria in order to relax convergence criteria for problems where one or few displacements dominate. In order to do that, use of ratio of Euclidean norm of iterative displacement \( \| \delta \mathbf{u} \|_{scaled} \) and Euclidean norm of total displacement \( \| \mathbf{u} \|_{scaled} \) is recommended:

\[
\frac{\| \delta \mathbf{u} \|_{relative}}{\| \mathbf{u} \|_{scaled}} \leq \epsilon_u \quad (107.31)
\]

Again it is important to note that currently used within F EI and the Real-ESSI program is an absolute convergence criteria with unit scaling matrix \( S = \mathbf{I} \). This means that relative tolerance criteria mixes units for different Degrees of Freedom (DoFs). Supplied tolerance is converted to basic units of the system (meter and Newton) and that is used for comparison and convergence decisions.

- **Average Displacement Convergence Criteria.** The change in the last correction \( \delta \mathbf{u} \) of the state vector \( \mathbf{u} \), measured in an appropriate norm, should not exceed a given tolerance \( \epsilon_u \), divided by the total number of DoFs:

\[
\| \delta \mathbf{u} \|_{average} = \frac{\sqrt{(\delta \mathbf{u})^T S (\delta \mathbf{u})}}{n \text{DoFs}} \leq \epsilon_u \quad (107.32)
\]

This is important in order to preserve objectivity of displacement convergence criteria for similar (same) models that are discretized with a different number of finite elements (and therefore feature different number of nodes and DoFs). For example, a cantilever (simplest model) can be discretized using 5 DoFs and 5,000,000 DoFs. Since equation 107.32 essentially sums up absolute values of all displacements, it is expected that in the case of larger number of DoFs, larger norm will be calculated. This will create a problem since specified tolerance will then be a function of a number of DoFs a model features. Hence, a norm of all iterative displacements is divided by the total number of DoFs.

One more time, it is important to note that currently used within F EI and the Real-ESSI program is an absolute convergence criteria with unit scaling matrix \( S = \mathbf{I} \). Therefor average tolerance criteria
mixes units. Supplied tolerance is converted to basic units of the system (meter and Newton) and that is used for comparison and convergence decisions.

- **Absolute Residual Force Convergence Criteria.** Since the residual $r(u, \lambda)$ measures the departure from the equilibrium path, an appropriate convergence test would be to compare Euclidean norm of residual with some predefined tolerance:

$$\|r(u, \lambda)\|_{scaled} = \sqrt{(r)^T S (r)} \leq \epsilon_r$$

(107.33)

Here, an obvious choice for scaling matrix is $S = diag(K_t)$

Much the same as for displacement based convergence criteria, currently used within FEI and the Real-ESSI program is a convergence criteria with unit scaling matrix $S = I$. This means that absolute tolerance criteria mixes units for different Degrees of Freedom (DoFs). Supplied tolerance is converted to basic units of the system (meter and Newton) and that is used for comparison and convergence decisions.

- **Relative Residual Force Convergence Criteria.** In order to provide scaling of residual forces that are used to tolerance criteria, (previously defined) absolute residual force norm is divided (scaled) by a a norm of residual forces at the beginning of the iterative step:

$$\|r(u, \lambda)\|_{relative} = \frac{\sqrt{(r)^T S (r)}}{\sqrt{(r_0)^T S (r_0)}} \leq \epsilon_r$$

(107.34)

Here, an obvious choice for scaling matrix is $S = diag(K_t)$, however, within FEI and the Real-ESSI program, for this convergence criteria, a unit scaling matrix $S = I$ is used, therefore relative tolerance criteria will feature mixed units.

- **Average Residual Force Convergence Criteria.** With the residual $r(u, \lambda)$, which measures the departure from the equilibrium path, there is a need to take into the account number of DoFs, in order to reduce the influence of a model discretization (number of DoFs) on norm of the residual force vector:

$$\|r(u, \lambda)\|_{scaled} = \frac{\sqrt{(r)^T S (r)}}{nDoFs} \leq \epsilon_{average}$$

(107.35)

Here, an obvious choice for scaling matrix is $S = diag(K_t)$ however, again, as noted above, within FEI and the Real-ESSI program, for this convergence criteria, a unit scaling matrix $S = I$ is used, therefore relative tolerance criteria will feature mixed units.
• **Energy Based Convergence Criteria.** The previous convergence criteria can be combined in a single work change criterion:

\[
\| (\delta u)^T (r) \| = \sqrt{ (\delta u)^T (r) } \leq \epsilon_u \epsilon_r \tag{107.36}
\]

A word of caution is appropriate at this point. As pointed out by Crisfield (1991), it follows that:

\[
\| (\delta u)^T (r) \| = \| (\delta u)^T (K_t^{-1} K_t) (r) \| = \| - (\delta u)^T K_t (\delta u) \| \leq \epsilon_u \epsilon_r \tag{107.37}
\]

where the iterative change was \((\delta u) = -K_t^{-1} r\). It should be noted that equations (107.37) give a measure of the "stiffness" of \(K_t\). This merely implies that a stationary energy position has been reached in the current iterative direction, \(\delta u\). This can occur when the solution is still far away from equilibrium.

Since \(u\) and \(r\) usually have physical units, so do necessarily \(\epsilon_u\) and \(\epsilon_r\). For a general purpose implementation of Newton–Raphson iteration this dependency on physical units is undesirable and it is more convenient to work with ratios that render the \(\epsilon_u\) and \(\epsilon_r\) dimensionless. **Displacement Convergence Criteria** can be rendered dimensionless by using ratio of scaled Euclidean norm of iterative displacement \(\| \delta u \|_{\text{scaled}} \) and scaled Euclidean norm of total displacement \(\| u \|_{\text{scaled}}\):

\[
\frac{\| \delta u \|_{\text{scaled}}}{\| u \|_{\text{scaled}}} \leq \epsilon_u \tag{107.38}
\]

The similar approach can be used for **Residual Convergence Criteria**:

\[
\frac{\| r \|_{\text{scaled}}}{\| r \|_{\text{predictor}}_{\text{scaled}}} \leq \epsilon_r \tag{107.39}
\]

Another important thing to be considered is **Divergence**. The Newton–Raphson scheme is not guaranteed to converge. Some sort of divergence detection scheme is therefore necessary in order to interrupt an erroneous iteration cycle. Divergence can be diagnosed if either of following inequalities occur:

\[
\frac{\| \delta u \|_{\text{scaled}}}{\| u \|_{\text{scaled}}} \geq g_u \tag{107.40}
\]

\[
\frac{\| r \|_{\text{scaled}}}{\| r \|_{\text{predictor}}_{\text{scaled}}} \geq g_r \tag{107.41}
\]

where \(g_u\) and \(g_r\) are **dangerous growth factors** that can be set to, for example \(g_u = g_r = 100\).
In some cases, the Newton–Raphson iteration scheme will neither diverge nor converge, but rather exhibit oscillatory behavior. To avoid excessive bouncing around, a good practice is to put upper limit to the number of iterations performed in one iteration cycle. Typical limits to the iteration number are 20 to 50.

### 107.6.5 The Algorithm Progress

The progress of the scheme will be briefly described, in relation with the Figure (107.1). The procedure starts from a previously converged solution \((u_0, \lambda_0 f_{ext})\). An incremental, tangential predictor step \(\Delta u_1, \Delta \lambda_1\) is obtained\(^{18}\) and the next point obtained is \((u_1, \lambda_1 f_{ext})\). The first iteration would then use quadratic equation 107.17 where constants \(a_1, a_2\) and \(a_3\) should be computed with \(\Delta u^{old} = \Delta u_1\) and \(\Delta \lambda^{old} = \Delta \lambda_1\), to calculate \(\delta \lambda_1\) and \(\delta u_1 = -K^{-1}_t \mathbf{r}(u_1, \lambda_1) + \delta \lambda_1 K^{-1}_t f_{ext}\). After these values are calculated, the updating procedure\(^{20}\) would lead to:

\[
\Delta \lambda_2 = \Delta \lambda_1 + \delta \lambda_1 \quad \text{and} \quad \Delta u_2 = \Delta u_1 + \delta u_1
\]

When added to the displacements \(u_0\) and load level \(\lambda_0\), at the end of the previous increment this process would lead to the next point \((u_2, \lambda_2 f_{ext})\).

The next iteration would then again use quadratic equation 107.17 where constants \(a_1, a_2\) and \(a_3\) should be computed with \(\Delta u^{old} = \Delta u_2\) and \(\Delta \lambda^{old} = \Delta \lambda_2\), to calculate \(\delta \lambda_3\) and \(\delta u_3 = \delta u + \delta \lambda_2 \delta u\). After these values are calculated, the updating procedure would lead to:

\[
\Delta \lambda_3 = \Delta \lambda_2 + \delta \lambda_2 \quad \text{and} \quad \Delta u_3 = \Delta u_2 + \delta u_2
\]

When added to the displacements \(u_0\) and load level \(\lambda_0\), at the end of the previous increment this process would lead to the next point \((u_3, \lambda_3 f_{ext})\).

---

\(^{18}\)As explained in Section (107.6.2).

\(^{19}\)From equation (107.12).

\(^{20}\)See (107.11) and (107.13)
Chapter 108

Solution of Dynamic Equations of Motion

(In collaboration with Dr. Nima Tafazzoli and Prof. José Abell)
108.1 Chapter Summary and Highlights

108.2 The Principle of Virtual Displacements in Dynamics

(see section 102.2 on page 93).

Great reading on this subject is a book by Argyris and Mlejnek (1991).

108.3 Direct Integration Methods for the Equations of Dynamic Equilibrium

This section follows Argyris and Mlejnek (1991) and Hughes (1987).

need to rewrite and improve! BJ

108.3.1 Newmark Integrator

The Newmark time integration method (Newmark, 1959) uses two parameters, \( \beta \) and \( \gamma \), and is defined by the following two equations:

\[
\begin{align*}
n^{+1}_x & = n_x + \Delta t \frac{1}{2} \left[ (1 - \beta) n_x + \beta n^{+1}_x \right] + \Delta t^2 \left[ (1 - \gamma) n_x + \gamma n^{+1}_x \right] \\
n^{+1}_\dot{x} & = n_\dot{x} + \Delta t \left[ (1 - \gamma) n_x + \gamma n^{+1}_x \right]
\end{align*}
\]  

These equations give the relationship between known variables at time step \( n \) to the unknown variables at next time step \( n + 1 \). Method is in general an implicit one, except when \( \beta = 0 \) and \( \gamma = 1/2 \).

There are several possible implementation methods for Newmark Integrator. One possible approach to integrating equations of motion using Newmark algorithm is to use displacement as the basic unknowns, and the following difference relations are used to relate \( n^{+1}_x \) and \( n^{+1}_\dot{x} \) to \( n_x \) and the response quantities are

\[
\begin{align*}
n^{+1}_\dot{x} & = \frac{\gamma}{\beta \Delta t} (n^{+1}_x - n_x) + \left( 1 - \frac{\gamma}{\beta} \right) n_\dot{x} + \left( 1 - \frac{\gamma}{2 \beta} \right) n_\ddot{x} \\
n^{+1}_\ddot{x} & = \frac{1}{\beta \Delta t^2} (n^{+1}_x - n_x) - \frac{1}{\beta \Delta t} n_\dot{x} + \left( 1 - \frac{1}{2 \beta} \right) n_\ddot{x}
\end{align*}
\]  

The predictors are then:

\[
\begin{align*}
n^{+1}_\dot{x}^\diamond & = -\frac{\gamma}{\beta \Delta t} n_x + \left( 1 - \frac{\gamma}{\beta} \right) n_\dot{x} + \Delta t \left( 1 - \frac{\gamma}{2 \beta} \right) n_\ddot{x} \\
n^{+1}_\ddot{x}^\diamond & = -\frac{1}{\beta \Delta t^2} n_x - \frac{1}{\beta \Delta t} n_\dot{x} + \left( 1 - \frac{1}{2 \beta} \right) n_\ddot{x}
\end{align*}
\]
and the correctors:

\[
\begin{align*}
    n+1\dot{x} &= n+1\dot{x}^0 + \frac{\gamma}{\beta \Delta t} n+1x \\
    n+1\ddot{x} &= n+1\ddot{x}^0 + \frac{1}{\beta \Delta t^2} n+1x
\end{align*}
\] (108.7) (108.8)

The Newton integration method becomes

\[
\begin{bmatrix} M & \frac{\gamma C}{\Delta t} + K \end{bmatrix} \Delta x = -n+1R
\] (108.9)

Equation 108.5 to 108.9 constitute an iterative solving procedure (Argyris and Mlejnek, 1991).

If the parameters \(\beta\) and \(\gamma\) satisfy

\[
\gamma \geq \frac{1}{2}, \quad \beta \geq \frac{1}{4}(\gamma + \frac{1}{2})^2
\] (108.10)

the procedure is unconditionally stable and second-order accurate. Any \(\gamma\) value greater than 0.5 will introduce numerical damping. Well-known members of the Newmark time integration method family include: trapezoidal rule or average acceleration method for \(\beta = 1/4\) and \(\gamma = 1/2\), linear acceleration method for \(\beta = 1/6\) and \(\gamma = 1/2\), and (explicit) central difference method for \(\beta = 0\) and \(\gamma = 1/2\). If and only if \(\gamma = 1/2\), the accuracy is second-order (Hughes, 1987). For values of \(\beta = 1\) and \(\gamma = 2/3\), the strongest numerical damping is obtained, as spectral ratio \(\rho_\infty = 0\) (Hughes (1987), page 502).

### 108.3.2 HHT Integrator

Numerical damping introduced in the Newmark time integration method will degrade the order of accuracy. The Hilber-Hughes-Tailor (HHT) time integration \(\alpha\)-method (Hilber et al., 1977), (Hughes and Liu, 1978a) and (Hughes and Liu, 1978b) using an alternative residual form by introducing an addition parameter \(\alpha\) to improve the performance:

\[
n+1R = M \ n+1\dot{x} + (1 + \alpha)F(n+1\dot{x},n+1x) - \alpha F(n\dot{x},nx) - n+1f
\] (108.11)

but retaining the Newmark finite-difference formulas 108.1 and 108.2 or 108.3 and 108.4. If \(\alpha = 0\), the HHT time integration method becomes exactly the same Newmark time integration method. Decreasing \(\alpha\) value increase numerical dissipation (Hughes, 1987).

The iteration method for HHT time integration is similar to that of Newmark time integration. Due to the change of Equation 108.11, Equation 108.9 becomes

\[
\begin{bmatrix} M + (1 + \alpha)\gamma \Delta t C + (1 + \alpha)\beta \Delta t^2 K \end{bmatrix} \Delta \ddot{x} = -n+1R
\] (108.12)

for acceleration iteration and

\[
\begin{bmatrix} \frac{1}{\beta \Delta t^2} M + \frac{(1 + \alpha)}{\gamma \Delta t} C + (1 + \alpha)K \end{bmatrix} \Delta x = -n+1R
\] (108.13)
for displacement iteration.

If the parameters $\alpha$, $\beta$ and $\gamma$ satisfy

$$-\frac{1}{3} \leq \alpha \leq 0, \quad \gamma = \frac{1}{2}(1 - 2\alpha), \quad \beta = \frac{1}{4}(1 - \alpha)^2$$

(108.14)

it is unconditionally stable and second-order accurate (Hughes, 1987).

## 108.4 Synthetic Viscous Damping for Solids and Structures

Presented here are commonly used, viscous damping methods for time domain analysis of solids and structures. These methods, Rayleigh and Caughey damping, are mimicking viscous damping of the solids and structures by generating the damping matrix $C$ using mass and stiffness matrices.

These synthetic viscous damping approaches should be distinguished from a natural viscous damping that is created during interaction of fluid and soil. For example, a natural viscous damping occurs when pore fluid and porous solid have differential displacements, as described in section 102.12.1.4 on page 137 (see for example equation ?? on page ??).

## 108.4.1 Synthetic Viscous Damping Approaches

There are different numerical methods available to simulate the seismic wave propagation through the soil-structure systems such as boundary elements, finite elements, finite differences, meshfree methods, and spectral elements. There are advantages of using methods such as finite elements or spectral elements for complex geometries or modeling the nonlinearities but also disadvantages such as numerical dispersion for low-order finite elements or reflection of the motions from the boundaries of the model (Semblat et al. (2010)).

Boundary element method can deal better with the issue of reflecting the motions from boundaries comparing to other numerical methods. Research has been done on coupling this method with other numerical methods for better applications. Domain reduction method is also available for large models implemented in finite element in order to reduce the problem of reflection (Bielak et al. (2003a)).

There are methods so called non-reflecting boundary conditions which directly can attenuate the reflections at the mesh boundaries. One of the commonly used method is absorbing boundary conditions. Absorbing boundary conditions have special conditions at the model boundaries in order to approximate the radiation condition for seismic waves (Givoli (1991)).

Another method applicable in finite element methods is so called infinite element method. These elements can absorb the waves using decaying laws at model boundaries at infinity (Nenning and Schanz (2010); Kallivokas et al. (1997)). In this method it is assumed that the element and nodes of the boundary
are in infinity. In this case the seismic waves have enough distance to dissipate at the boundaries and not to reflect back to the model.

There has been a recently developed method to prevent reflection of the waves from boundaries called Perfectly Matched Layers (PML). This absorbing layer is based on attenuation laws with specific properties and finite thickness located at the model boundaries. There are several PML formulations proposed for finite element methods which allows the treatment of surface waves as well as body waves (Festa and Nielsen (2003); Basu (2009)).

Bilbao et al. (2006) proposed two energy-based methods to model damping in structures with added viscoelastic dampers. These methods approximate the effects of the added viscoelastic dampers with a damping matrix in the form of Caughey damping matrix.

The method to be used here is so called Caughey Absorbing Layer Method. The 2nd order of this method is also known as a Rayleigh damping. Caughey damping is a classical method in which the damping matrix is built based on the mass and stiffness matrices. Since the stiffness and mass matrices of have to be created for solving the system of equations, they are used for creating the damping matrix as well.

Considering the relationship between internal friction and frequency for damping, it is possible to build a model involving the same attenuation/frequency dependence for Caughey damping (Semblat (1997)). The relation of the inverse of the quality factor $Q^{-1}$ and the damping ratio $\xi$ can be written as:

$$Q^{-1} \approx 2\xi$$  \hspace{1cm} (108.15)

Caughey damping formulation in general can be expressed as

$$C = [M] \sum_{j=0}^{m-1} a_j ([M]^{-1}[K])^j$$  \hspace{1cm} (108.16)

where the order to be used depends on number of modes to be considered for damping in the problem.

The way it is implemented in ESSI Simulator gives the opportunity to the user to use different types of damping for different elements. There might not be a need to use damping for all the elements of the model. In this case, damping could be used for particular elements and leave the rest of them with no physical Caughey damping. It can also be used for damping out the residual waves coming out of the domain reduction boundary layer.
108.4.2 Caughey Damping 2nd Order, aka Rayleigh Damping

The second order Caughey damping, is also known as a Rayleigh damping, with $j = 1$ in Equation (108.16). From dynamic parameters and formulation of the system following equations can be observed:

\[ \xi_n = \frac{C_n}{2M_n\omega_n} \]  
\[ (108.17) \]

\[ K_n = \omega_n^2 M_n \]  
\[ (108.18) \]

Considering the first two terms in Caughey damping formulation, if the damping matrix formulation is written separately for each term:

\[ C = a_0 M \]  
\[ (108.19) \]

Based on Equation (108.17), $a_0$ can be written as:

\[ a_0 = 2\xi\omega \]  
\[ (108.20) \]

Writing the damping matrix based on the second coefficient:

\[ C = a_1 K = a_1\omega^2 M \]  
\[ (108.21) \]

Then $a_1$ can be obtained as:

\[ a_1 = \frac{2\xi}{\omega} \]  
\[ (108.22) \]

So the damping ratio of the $n^{th}$ mode of the system is:

\[ \xi_n = \frac{a_0}{2} \frac{1}{\omega_n} + \frac{a_1}{2} \omega_n \]  
\[ (108.23) \]

Presenting Equation (108.23) for first two modes in matrix form leads to:

\[ \frac{1}{2} \begin{bmatrix} \frac{1}{\omega_i} & \omega_i \\ \frac{1}{\omega_j} & \omega_j \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \xi_i \\ \xi_j \end{bmatrix} \]  
\[ (108.24) \]
The following procedure given by Hall (2006) is useful to conveniently determine Rayleigh damping coefficients $a_0$ and $a_1$. Select a desired amount of damping $\xi$ and a frequency range from $\hat{\omega}$ to $R\hat{\omega}$, where $R > 1$. Compute $\Delta$ from:

$$\Delta = \xi \frac{1 + R - 2\sqrt{R}}{1 + R + 2\sqrt{R}}$$

(108.25)

where $\Delta$ determines bounds on the damping ratios that are imparted to those modes within the specified frequency range. Any such mode will have a damping ratio bounded by $\xi_{\text{max}} = \xi + \Delta$ and $\xi_{\text{min}} = \xi - \Delta$. If these bounds are satisfactorily narrow, the constants $a_0$ and $a_1$ are then calculated from:

$$a_0 = 2\xi \hat{\omega} \frac{2R}{1 + R + 2\sqrt{R}}$$

(108.26)

$$a_1 = 2\xi \frac{1}{\omega} \frac{2}{1 + R + 2\sqrt{R}}$$

(108.27)

and can be used to compute an actual damping value $\xi_n$ for mode $n$ from Equation (108.23).

Figure 108.1: Actual damping ratio $\xi_n$ of mode $n$ as a function of frequency $\omega_n$ of mode $n$ when using Rayleigh damping (Hall (2006)).

Figure (108.1) shows that $\xi_n = \xi_{\text{max}}$ if $\omega_n = \hat{\omega}$ or $\omega_n = R\hat{\omega}$, and that $\xi_n = \xi_{\text{min}}$ if $\omega_n = \sqrt{R}\hat{\omega}$. If $\omega_n$ is outside the range $\hat{\omega}$ to $R\hat{\omega}$, then $\xi_n > \xi_{\text{max}}$. Above $R\hat{\omega}$, $\xi_n$ increases with $\omega_n$, approaching a linear relation as the last term in Equation (108.23) dominates.
Note: It’s worth pointing out that damping could be unrealistically high for motions outside the prescribed frequency range, if the frequency range is not well-chosen.

108.4.3 Caughey Damping 3rd Order

Following the same logic as the 2nd order, the last coefficient of the 3rd order Caughey damping formulation can be written as following:

\[
C = a_2 KM^{-1} K = a_2 \omega^4 M
\]  

(108.28)

So \( a_2 \) can be obtained as:

\[
a_2 = \frac{2\xi}{\omega^3}
\]  

(108.29)

Considering the last coefficient in the formulation, the damping ratio of the system can now be shown as:

\[
\xi_n = \frac{a_0}{2} \frac{1}{\omega_n} + \frac{a_1}{2} \omega_n + \frac{a_2}{2} \omega^3_n
\]  

(108.30)

By solving the following set of equations, 3rd order Caughey damping coefficients can be found:

\[
\begin{bmatrix}
\frac{1}{\omega_i} & \omega_i^3 \\
\frac{1}{\omega_j} & \omega_j^3 \\
\frac{1}{\omega_k} & \omega_k^3
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2
\end{bmatrix}
= 
\begin{bmatrix}
\xi_i \\
\xi_j \\
\xi_k
\end{bmatrix}
\]  

(108.31)

108.4.4 Caughey Damping 4th Order

The 4th coefficient of the Caughey damping formulation can be obtained as:

\[
C = a_3 KM^{-1} K M^{-1} K = a_3 \omega^6 M
\]  

(108.32)

\[
a_3 = \frac{2\xi}{\omega^5}
\]  

(108.33)
\[ \xi_n = \frac{a_0}{2} \frac{1}{\omega_n} + \frac{a_1}{2} \omega_n + \frac{a_2}{2} \omega_n^3 + \frac{a_3}{2} \omega_n^5 \] (108.34)

So the damping coefficients can be obtained by solving the following set of equations:

\[
\begin{bmatrix}
\frac{1}{\omega_i} & \omega_i^3 & \omega_i^5 \\
\frac{1}{\omega_j} & \omega_j^3 & \omega_j^5 \\
\frac{1}{\omega_k} & \omega_k^3 & \omega_k^5 \\
\frac{1}{\omega_l} & \omega_l^3 & \omega_l^5 \\
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
\xi_i \\
\xi_j \\
\xi_k \\
\xi_l \\
\end{bmatrix}
\] (108.35)
Chapter 109

Earthquake Soil Structure Interaction, Theoretical Aspects


(In collaboration with Dr. Nima Tafazzoli, Prof. José Abell, Dr. Yuan Feng, Dr. Hexiang Wang)
109.1 Chapter Summary and Highlights

109.2 Seismic Energy Propagation and Dissipation

Jeremić (2010)

109.2.1 Seismic energy input into SSI system

Earthquakes release large amounts of energy at the source\(^1\) Part of released energy is radiated as mechanical waves (\(\approx 1.6 \times 10^{-5}\)) and part of that energy makes it to the surface where SSI system is located.

Mechanical seismic wave energy enters the SSI system through a closed surface \(\Gamma\) that encompasses (significant) soil volume as well as foundation system and the structure (see Figure 109.7). Kinetic energy flux through closed surface \(\Gamma\) includes both incoming and outgoing waves and can be calculated using Domain Reduction Method (Bielak et al., 2003a) as:

\[
E_{\text{flux}} = \left[ 0; -M^{\Omega+}_{be} \dot{u}^0_e - K^{\Omega+}_{be} u^0_e; M^{\Omega+}_{eb} \ddot{u}^0_b + K^{\Omega+}_{eb} u^0_b \right]_i \times u_i
\]

where \(M^{\Omega+}_{be}, M^{\Omega+}_{eb}, K^{\Omega+}_{be}, K^{\Omega+}_{eb}\) are mass and stiffness matrices, respectively for a single layer of elements just outside of the boundary \(\Gamma\), while \(\ddot{u}^0_e\) and \(u^0_e\) are accelerations and displacements from a free field model for nodes belonging to that layer of elements. Alternatively, energy flux can be calculated using ((Aki and Richards, 2002), page 122):

\[
E_{\text{flux}} = \rho A c \int_0^t \ddot{u}_i^2 dt
\]

Outgoing kinetic energy can be obtained from outgoing wave field \(w_i\), (from DRM, Bielak et al. (2003a)), while the difference then represents the incoming kinetic energy that needs to be dissipated with SSI region.

109.2.2 Seismic Energy Dissipation in SSI System

Seismic energy that enters the SSI system will be dissipated in a number of ways. Part of the energy that enters SSI system is reflected and radiated back into domain outside \(\Gamma\) by

- wave reflection from impedance boundaries (free surface, soil/rock layers, foundations, etc.).
- SSI system oscillates and emits, radiates waves back into the domain

---

\(^1\) for example, some of the recent large earthquake energy releases are listed: Northridge, 1994, \(M_{\text{Richter}} = 6.7, E_r = 6.8 \times 10^{16} J\); Loma Prieta, 1989, \(M_{\text{Richter}} = 6.9, E_r = 1.1 \times 10^{17} J\); Sumatra-Andaman, 2004, \(M_{\text{Richter}} = 9.3, E_r = 4.8 \times 10^{20} J\); Valdivia, Chile, 1960, \(M_{\text{Richter}} = 9.5, E_r = 7.5 \times 10^{20} J\);
The rest of seismic energy is dissipated through one of the following mechanisms within SSI system:

- Inelastic, elasto-plastic behavior of soil and rock
- Inelastic, elasto-plastic, damage behavior of the foundation system
- Inelastic, elasto-plastic, damage behavior of the structure
- Viscous coupling of porous solid with pore fluid (air, water)
- Viscous coupling of structure with surrounding, internal and external fluids (air, water)

It is also important to note that in numerical simulations, part of the energy can be dissipated or produced by purely numerical means. That is, numerical energy dissipation (damping) or production (negative damping) has to be carefully controlled (Argyris and Mlejnek, 1991), (Hughes, 1987).

**109.2.2.1 Energy Dissipation by Plasticity**

Elastic-plastic deformation of soil, foundation and structure is probably responsible for major part of the energy dissipation for large earthquakes. This, displacement proportional dissipation is a result of plastic dissipation and is present in all three components of the system (soil, foundation and the structure).

A note about plastic dissipation is important at this point. There is a misconception about plastic energy dissipation that is being widely used. Here are some details:

- **Origins of the Misconception:** The paper by Uang and Bertero (1990) has been considered the definitive work in using energy as a measure of structural demand by many researchers (Léger and Dussault, 1992; Cosenza et al., 1993; Kalkan and Kunnath, 2007, 2008; Symans et al., 2008; Gajan and Saravanathiiban, 2011; Moustafa, 2011; Moustafa and Mahmoud, 2014; Mezgebo and Lui, 2017; Deniz et al., 2017). An energy analysis methodology based on absolute input energy (or energy demand) was presented and discussed. Numerical analysis results were compared with experiments on a multi-story building. In this paper, hysteretic energy is calculated indirectly by taking the difference of absorbed energy and elastic strain energy. The term absorbed energy of each time step was simply defined as force times incremental displacement. It was stated that hysteretic energy is irrecoverable, which indicates that this parameter was considered the same as hysteretic dissipation or plastic dissipation. An equation for energy balance is given:

\[
E_i = E_k + E_\xi + E_a = E_k + E_\xi + E_s + E_h
\]

where \(E_i\) is the (absolute) input energy, \(E_k\) is the (absolute) kinetic energy, \(E_\xi\) is the viscous damping energy, \(E_a\) is the absorbed energy, which is composed of elastic strain energy \(E_s\) and hysteretic energy \(E_h\).
The problem of this theory is the absence of plastic free energy, which is necessary to correctly evaluate energy dissipation of elastic-plastic materials and to uphold the second law of thermodynamics. There was no direct plot of plastic dissipation (hysteretic energy) in this paper, since it was not defined directly. There were plots of other energy components and plastic dissipation can be easily calculated or deducted from these plots. After doing this, indications of negative incremental energy dissipation, which violates the basic principles of thermodynamics, were found in various sections of the paper.

This misconception could be clarified by renaming hysteretic energy to plastic work, which is the combination of plastic dissipation and plastic free energy. Both plastic work and plastic free energy can be incrementally negative, but plastic dissipation (defined as the difference of plastic work and plastic free energy) must be incrementally non-negative during any time period. Unfortunately, this misconception has been inherited (if not magnified) by almost all following studies on energy analysis of earthquake soils and structures.

Besides, another issue regarding energy dissipation is found in this paper. Viscous damping energy (or viscous damping energy) was calculated directly using damping coefficient and velocity. The author stated that this term should always be non-negative. But it was ignored that the incremental viscous energy dissipation should also be non-negative. In fact, the equation used to compute viscous energy dissipation should be able to ensure that it remains non-negative incrementally. However, it appeared in one of the plots that (accumulated) viscous damping energy was clearly dropping during certain time periods, which was in contradiction with the equation derived in the same paper. Such result was also a violation of thermodynamics.

- **Misconception in Other Studies:** Although input energy was the key parameter used in Uang and Bertero (1990), the misconception of energy dissipation has been carried on in a vast number of studies on energy analysis of ESSI systems.

Léger and Dussault (1992) used Equation 109.1 from Uang and Bertero (1990) to perform energy response analysis of multi-story buildings under earthquake loading. It was stated that the total input energy at the end of ground motion is approximately equal to the total dissipated energy. This statement is only valid if plastic free energy remains constant, which is generally not true in elastic-plastic materials.

Kalkan and Kunnath (2007) calculated energy dissipation of a single-degree-of-freedom (SDOF) oscillator under earthquake loading. Inelastic material was used so plastic energy dissipation appeared in the results. Negative incremental hysteretic energy (plastic dissipation) was observed in the plots, which is a clear violation of thermodynamics. The change of plastic free energy was not
considered in this study. Several papers by the same authors performed energy analysis on various structures using the same theory, and similar misconceptions can be noticed in these publications.

Symans et al. (2008) summarized current practice and recent developments in the application of passive energy dissipation systems for seismic protection of structures. There was no consideration of plastic free energy in the energy balance equation, which was very close to the ones present by Uang and Bertero (1990) and Léger and Dussault (1992). It was stated that the cumulative hysteretic energy is equal to the energy demand (absolute input energy) at the end of earthquake. Although no direct violation of thermodynamics was observed, such statement clearly indicated the same misconception of plastic work and plastic dissipation appeared in other publications.

Gajan and Saravanathiiban (2011) performed both numerical simulations and centrifuge experiments on a rocking foundation system. Energy dissipation in foundation soil and structural elements were calculated. It can be observed that hysteretic energy dissipation in both the soil and the structure was decreasing during certain time periods, which is a direct violation of thermodynamics. Again, this was a misconception of plastic work and plastic energy dissipation. The change in plastic free energy of the system was significant in this case, since large drops of plastic work were noticed in the plots.

A number of recent studies Moustafa (2011); Moustafa and Mahmoud (2014); Mezgebo and Lui (2017); Deniz et al. (2017) performed energy analysis on ESSI systems without considering plastic free energy. Misleading results were obtained using the wrong energy balance equation. This means that the misconception of plastic work and plastic energy dissipation is still not realized by many researchers. Note that the influence of this mistake could be negligible or significant, depending on the case analyzed. Nevertheless, plastic free energy should not be ignored without plausible reasoning or experimental evidences.

- **Early Studies on Plastic Free Energy (Cold Work):** This issue has been pointed out and studied extensively by researchers from mechanical engineering and material science. In the early 20th century, Taylor and his colleagues Farren and Taylor (1925), Taylor and Quinney (1934) performed experiments on metals and proved that a large part, but not all, of the input mechanical energy is converted into heat. The remaining part of the non-recoverable plastic work is known as the stored energy of cold work. The ratio of plastic work converted into heating (Quinney–Taylor coefficient), usually denoted as $\beta$, has been used in almost all later papers on this topic. Based on large amount of experimental data, this ratio was assumed to be a constant between 0.6 to 1.0 in many studies (e.g. Clifton et al. (1984), Belytschko et al. (1991), Zhou et al. (1996), Dolinski et al. (2010), Ren and Li (2010), Osovski et al. (2013)). It has been realized that this assumption
is not valid in all cases, but it’s simply too complicated to involve the evolution of Quinney–Taylor coefficient in thermomechanical constitutive models.

Decades later, Mason et al. (1994) showed that the fraction of plastic work converted into heat is both strain and strain rate dependent. Infrared imaging was used in this study (and almost all future studies) to obtain temperature distribution in the material, because it is the only effective approach to directly measure energy dissipation (heat). In the recent 20 years, there has been many developments on this issue. Rittel (Rittel, 2000; Rittel and Rabin, 2000; Rittel et al., 2003) published several insightful papers on the energy dissipation (heat generation) of polymers during cyclic loading, presenting both experimental and theoretical works. Rosakis et al. (2000) presented a constitutive model based on thermoplasticity to model the evolution of in metal. This model is capable of calculating the evolution of energy dissipation and material properties, and is validated by sets of experiments. There are some follow-up papers by Rosakis and his colleagues (Hodowany et al., 2000; Ravichandran et al., 2002) on the same issue with some assumptions to simply the problem. One widely used assumption is the adiabatic condition, since air conducts heat much slower than metal. This assumption is reasonable in rapid monotonic or cyclic loading (impact, vibration, earthquake). One application of this theory in geotechnical engineering is presented in papers of (Veveakis et al., 2007, 2012), in which thermoporomechanics is used to model the heating and pore pressure increase in large landslides, like the 1963 Vajont slide in Italy.

In the field of civil engineering, the basic principles of thermodynamics are frequently used to derive new constitutive models Dafalias and Popov (1975); Ziegler and Wehrli (1987); Collins and Houlsby (1997); Houlsby and Puzrin (2000); Collins (2002); Collins and Kelly (2002); Collins (2003); Feigenbaum and Dafalias (2007). to enforce the second law of thermodynamics for developed constitutive models. Note that plastic free energy is the same concept as cold work. The former term is more popular in solid physics and geotechnical engineering, while the latter is used in mechanical engineering. energy dissipation due to plasticity and plastic work, which is often a source of a confusion. through a conceptual example that is analyzed on particle scale. development of plastic free energy is caused by particle rearrangement in granular assembly under external loading.

Ideally, majority of the incoming energy would be dissipated in soil, before reaching foundation and structures. The possibility to direct energy dissipation to soil can be used in design by recognizing energy dissipation capacity for different soils. For example, simple elastic-plastic models of stiff and soft clay as well as dense and loose sand predict different energy dissipation capacities, as shown in Figure 109.1, for single loading-unloading-reloading cycle. While Figure 109.1 shows that stiff clay and dense sand have
109.2.2 Energy Dissipation by Viscous Coupling

Viscous coupling of pore fluid (air, water...) and soil particles and/or foundation or structural components is responsible for velocity proportional energy dissipation. In particular, viscous coupling of porous solid and fluid results in $E_{vc} = n^2 k^{-1} (\dot{U}_i - \dot{u}_i)^2$ energy loss per unit volume. It is noted that this type of dissipation is realistically modeled using $u - p - U$ formulation (Jeremić et al., 2008).

109.2.2.3 Numerical Energy Dissipation and Production

As noted above, numerical integration of nonlinear equations of motions affects calculated energy in various ways. Most common effect for nonlinear (elastic-plastic) systems is the positive (energy dissipation) and negative (energy production) damping. For example Newmark (N) (Newmark, 1959) and Hilber–Hughes–Taylor (HHT) (Hilber et al., 1977) are energy preserving for linear elastic system with proper choice of constants ($\alpha = 0.0; \beta = 0.25, \gamma = 0.5$). Both methods can also be used to dissipate higher frequency modes for linear elastic models by changing constants so that for N: $\gamma \geq 0.5, \beta = 0.25(\gamma+0.5)^2$, while for HHT: $-0.33 \leq \alpha \leq 0, \gamma = 0.5(1-2\alpha), \beta = 0.25(1-\alpha)^2$. However, for nonlinear problems it is impossible to maintain energy of the system throughout computations (Argyris and Mlejnek, 1991).

109.2.2.4 Energy Dissipation by Nonlinearities in Soil/Rock

Elasto-plasticity of solid skeleton

Viscous (coupling) effects
109.2.2.5 Energy Dissipation by Nonlinearities in Soil/Rock – Foundation Interface Zone

- Gap, Slip
  - Dry
  - Saturated

109.2.2.6 Energy Dissipation by Nonlinearities in Seismic Isolators

109.2.2.7 Energy Dissipation by Nonlinearities in Structures, Systems and Components

109.2.2.8 Numerical Energy Dissipation and Production

109.2.3 Seismic Motions: Empirical Models
109.2.4 1D/1C Wave Propagation Modeling

The theory of wave propagation is associated with vertical propagation of shear and/or compresional waves through the linear viscoelastic system is described in this section. Developments shown here follow standard approach, as found by, for example, Kramer (1996a). Shown in Figure 109.2 is a 1C wave propagation setup. The model consists of N horizontal layers, that extend to infinity in the horizontal direction. At the bottom of layers is bedrock that represents a halfspace. Each layer is homogeneous and isotropic and is characterized by the thickness h, mass density $\rho$, shear wave velocity $V_s^2$, and compressional wave velocity $V_p^3$ and damping factor, $\beta$.

![Wave Propagation Diagram]

Figure 109.2: Problem Description: Wave Propagation

Vertical propagation of shear or compressional waves will cause only horizontal or vertical displacements respectively.

$$u = u(z, t)$$  \hspace{1cm} (109.2)

Focusing on vertical propagation of shear waves, and with a presence of displacements in $x$ direction only, a wave equation, that describes wave propagation in vertical, $z$ directions can be written as

$$\rho \frac{\partial^2 u}{\partial t^2} = G \frac{\partial^2 u}{\partial z^2} + \eta \frac{\partial^3 u}{\partial z^2 \partial t}$$  \hspace{1cm} (109.3)

Shear Wave velocity is used to obtain shear shear modulus $G$, $V_s = \sqrt{G/\rho}$, $G = V_s^2 \rho$, $G = E/(2(1+\nu))$, $E = (9KG)/(3K + g)$, $E = (G(3M - 4G))/(M - G)$.

Compressional wave velocity is used to obtain constrained modulus $M$, $V_p = \sqrt{M/\rho}$, $M = V_p^2 \rho$, $M = K + 4G/3$, $M = E(1-\nu)/((1+\nu)(1-2\nu))$. 

---

*Jeremić et al.* University of California, Davis  
Harmonic oscillation, displacements with frequency $\omega$, can be written in the as:

$$u(z, t) = U(z) \cdot e^{i\omega t}$$  \hspace{1cm} (109.4)

Substituting Eq (109.4) into Eq (109.3) one obtains

$$(G + i\omega \eta) \frac{\partial^2 u}{\partial z^2} = \rho \omega^2 U$$  \hspace{1cm} (109.5)

which has the general solution:

$$U(z) = E e^{ikz} + F e^{-ikz}$$  \hspace{1cm} (109.6)

in which

$$k^2 = \frac{\rho \omega^2}{G + i\omega \eta} = \frac{\rho \omega^2}{G^*}$$  \hspace{1cm} (109.7)

where $k$ is the complex wave number and $G^*$ is the complex shear modulus. The critical damping ratio, $\beta$, is related to the viscosity $\eta$ by

$$\omega \eta = 2G \beta$$  \hspace{1cm} (109.8)

For convenience, one can use use soil damping ratio $\beta$ to represent the complex shear modulus.

$$G^* = G + i\omega \eta = G(1 + 2i\beta)$$  \hspace{1cm} (109.9)

By combine Eq. (109.4) and Eq. (109.6) one obtains the wave equation for a harmonic motion of frequency $\omega$.

$$u(z, t) = E e^{i(kz + \omega t)} + F e^{-i(kz - \omega t)}$$  \hspace{1cm} (109.10)

where the first term represents the incident wave traveling in the upward, in positive $z$ direction and the second term represents the reflected wave traveling in the negative, downward $z$–direction.

This equation is valid for each of the soil and rock layers.

Introducing a local coordinate system $Z$ for each layer, the displacements at the top and bottom of layer $m$ are:

$$u_m(z = 0) = (E_m + F_m)e^{i\omega t}$$  \hspace{1cm} (109.11)

$$u_m(z = h_m) = (E_m e^{ik_m h_m} + F_m e^{-ik_m h_m}) \cdot e^{i\omega t}$$  \hspace{1cm} (109.12)
The shear stress on a horizontal plane is
\[ \tau(z, t) = G \cdot \frac{\partial u}{\partial z} + \eta \frac{\partial u}{\partial z \partial t} = G^* \frac{\partial u}{\partial z} \] (109.13)

In another form,
\[ \tau(z, t) = ikG^*(Ee^{ikz} - Fe^{-ikz})e^{i\omega t} \] (109.14)

and the shear stress at the top and bottom of layer \( m \) are respectively:
\[ \tau_m(z = 0) = ik_m G^m(E_m - F_m)e^{i\omega t} \] (109.15)
\[ \tau_m(z = h_m) = ik_m G^m(E e^{ik_m h_m} - F e^{-ik_m h_m})e^{i\omega t} \] (109.16)

Stresses and displacements must be continuous at all interfaces. Hence, by Eq (109.11), (109.12), (109.15) and (109.16), the coefficients are
\[ E_{m+1} + F_{m+1} = E_m e^{ik_m h_m} + F_m e^{-ik_m h_m} \] (109.17)
\[ E_{m+1} - F_{m+1} = \frac{k_m G_m^*}{k_{m+1} G_{m+1}^*} (E_m e^{ik_m h_m} - F_m e^{-ik_m h_m}) \] (109.18)

Subtraction and addition Eqs. 15 and 16 yield the following recursion formulas for the amplitudes, \( E_{m+1} \) and \( F_{m+1} \), of the incident and reflected wave in layer \( m+1 \), expressed in terms of the amplitudes in layer \( m \):
\[ E_{m+1} = \frac{1}{2} E_m (1 + \alpha_m) e^{ik_m h_m} + \frac{1}{2} F_m (1 - \alpha_m) e^{-ik_m h_m} \] (109.19)
\[ F_{m+1} = \frac{1}{2} E_m (1 - \alpha_m) e^{ik_m h_m} + \frac{1}{2} F_m (1 + \alpha_m) e^{-ik_m h_m} \] (109.20)

where \( \alpha_m \) is the complex impedance ratio
\[ \alpha_m = \frac{k_m G_m^*}{k_{m+1} G_{m+1}^*} = \left( \frac{\rho_m G_m^*}{\rho_{m+1} G_{m+1}^*} \right)^{1/2} \] (109.21)

At the free surface, the shear stresses must be zero. In addition, according to Eq. (109.15), it obtains \( E_1 = F_1 \). Namely, the amplitudes of the incident and reflected waves are always equal at the free
surface. Beginning with the surface layer, repeated use of Eq (109.19) and Eq (109.20) to build the wave field:

\[ E_m = e_m(\omega)E_1 \]  
\[ F_m = f_m(\omega)E_1 \]  

(109.22)  

(109.23)

The transfer function \( e_m \) and \( f_m \) are simply the amplitudes for the case \( E_1 = F_1 = 1 \), and can be determined by substituting this condition into the above recursion formulas. Other transfer functions are easily obtained from the \( e_m \) and \( f_m \) functions. The transfer function \( A_{n,m} \) between the displacements at level \( n \) and \( m \) is defined by:

\[ A_{n,m}(\omega) = \frac{u_m}{u_n} \]  

(109.24)

and can be represented by

\[ A_{n,m}(\omega) = \frac{e_m(\omega) + f_m(\omega)}{e_n(\omega) + f_n(\omega)} \]  

(109.25)

Based on these equations, the transfer function \( A(\omega) \) can be found between any two layers in the system. Hence, if the motion is known in any one layer in the system, the motion can be computed in any other layer.

In summary, after the \( E \) and \( F \) are computed for all layers in the system, the accelerations are expressed by the equation:

\[ \ddot{u}(z,t) = \frac{\partial^2 u}{\partial t^2} = -\omega^2(Ee^{i(\kappa z + \omega t)} + Fe^{-i(\kappa z - \omega t)}) \]  

(109.26)

### 109.2.5 Seismic Motions: 3D/3C Analytic Wave Propagation Modeling

This is based in part on Wang et al. (2020).

Considered is the inclined wave propagation in the layered ground as shown in Fig. 109.3. There are \( n \) layers with layer thickness \( d_m \), density \( \rho_m \), compressional velocity \( \alpha_m \) and shear wave velocity \( \beta_m \) \( (m = 1, 2, ..., n) \). Since the incidence of out-of-plane SH wave is simpler (no mode conversion), here we focus on the incidence of P and SV wave and account for the mode conversion between them. The wave potential formulations below are general and also applicable to incident SH wave (Haskell, 1953). Without loss of generality, incident waves is considered to be monochromatic with angular frequency \( \omega \) and horizontal phase velocity \( c \). For incident waves with arbitrary time signal and multiple frequencies, free field motions can be Fourier synthesized from the monochromatic solutions.
According to Helmholtz decomposition theorem (Arfken and Weber, 1999), the displacement of wave propagation Eq. (109.27) in linear elastic media can be expressed with $P$ wave scalar potential $\phi$ and $S$ wave vector potential $\Psi$ as shown in Eq. 109.28, where $\phi$ is the curl free part corresponding to volumetric deformation and $\Psi$ is divergence free part corresponding to deviatoric deformation.

$$\rho \ddot{u} = (\lambda + 2\mu) \nabla \nabla \cdot u - \mu \nabla \times \nabla \times u$$ (109.27)

$$u = \nabla \phi + \nabla \times \Psi$$ (109.28)

Therefore, the unknown variables for $m^{th}$ layer are simplified into incident $P$ wave potential magnitude $\phi'_m$, reflected $P$ wave potential magnitude $\phi''_m$, incident $SV$ wave potential magnitude $\Psi'_m$ and reflected $SV$ wave potential magnitude $\Psi''_m$.

$$\phi_m = [\phi'_me^{ik(x-\gamma_mz)} + \phi''_me^{-ik(x+\gamma_mz)}]e^{-iwt}$$

$$\Psi_m = [\Psi'_me^{ik(x-\gamma_mz)} + \Psi''_me^{-ik(x+\gamma_mz)}]e^{-iwt}$$ (109.29)

The $P$ and $SV$ wave potential can be expressed as Eq. (109.29), where $k$ is the horizontal wave number, equals to $w/c$. And $\cot^{-1}\gamma_m$ and $\cot^{-1}\gamma_m$ are incident and reflected angles for $P$ and $SV$ waves.
wave, respectively. The harmonic nature of the potential field is characterized by the time factor $e^{-iwt}$. It will be understood and omitted hereafter. $\gamma_m$ and $\beta_m$ can be determined by Snell’s law (equation (??)). Note that when $\alpha_m$ or $\beta_m$ is greater than $c$, the incidence from $P$ or $SV$ wave is beyond the critical angle. $\gamma_m$ and $\beta_m$ become complex numbers. The plane wave magnitude exponentially decays along the depth. To be consistent with the original formulation by Haskell (1953), dilatational wave solutions $\Delta_m$ and rotational wave solutions $\omega_m$ are first introduced as Eq. (109.30).

\[
\Delta = \frac{\partial u_x}{\partial x} + \frac{\partial u_z}{\partial z} \\
\omega = \frac{1}{2} \left( \frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x} \right)
\]

(109.30)

$\phi_m$ and $\Psi_m$ are related to $\Delta_m$ and $\omega_m$ as follows:

\[
\phi_m = -\frac{(\alpha_m)^2}{\omega} \Delta_m \\
\Psi_m = 2(\beta_m)^2 \omega_m
\]

(109.31)

The displacements $(u_x, u_y)$ and interfacial stresses $(\sigma_{zz}, \tau_{zx})$ can be expressed in wave potential magnitudes $\phi$ and $\Psi$.

From Eqs. (109.28)-(109.31), the displacement and stress field of $m^{th}$ layer can be calculated from the dilatational wave and rotational wave solutions $\Delta_m$ and $\omega_m$ as follows:

\[
u_x = \{-ik\frac{(\alpha_m)^2}{\omega}[(\Delta'_m + \Delta''_m)\cos(k\gamma_m z) - i(\Delta'_m - \Delta''_m)\sin(k\gamma_m z)]
+2ik\gamma_m\frac{(\beta_m)^2}{\omega}[(\omega'_m - \omega_m)\cos(k\beta_m z) + i(\omega'_m + \omega_m)\sin(k\beta_m z)]\}e^{ikx}
\]

(109.32)

\[
u_z = \{ik\gamma_m\frac{(\alpha_m)^2}{\omega}[(\Delta'_m - \Delta''_m)\cos(k\gamma_m z) - i(\Delta''_m + \Delta'_m)\sin(k\gamma_m z)]
+2ik\frac{(\beta_m)^2}{\omega}[(\omega'_m + \omega_m)\cos(k\beta_m z) - i(\omega'_m - \omega_m)\sin(k\beta_m z)]\}e^{ikx}
\]

(109.33)

\[
\sigma_{zz} = \rho_m\{\alpha_m^2(1 - 2\frac{(\beta_m)^2}{c^2})[(\Delta'_m + \Delta''_m)\cos(k\gamma_m z) - i(\Delta'_m - \Delta''_m)\sin(k\gamma_m z)]
+4\frac{(\beta_m)^4}{c^2}\gamma_m[(\omega'_m - \omega_m)\cos(k\beta_m z) - i(\omega'_m + \omega_m)\sin(k\beta_m z)]\}e^{ikx}
\]

(109.34)

\[
\tau_{zx} = 2\rho_m\beta_m\frac{(\alpha_m)^2}{c}[((\Delta'_m - \Delta''_m)\cos(k\gamma_m z) - i(\Delta''_m + \Delta'_m)\sin(k\gamma_m z)]
+[(1 - 2\frac{(\beta_m)^2}{c^2})[(\omega'_m + \omega_m)\cos(k\beta_m z) - i(\omega'_m - \omega_m)\sin(k\beta_m z)]\}e^{ikx}
\]

(109.35)
Define the displacement and stress solution at \( m \)th interface as \( S^{(m)} \), which is equal to \([\ddot{u}_x(z_m = d_m)/c, \ddot{u}_z(z_m = d_m)/c, \sigma_{zz}(z_m = d_m), \tau_{zx}(z_m = d_m)]^T\). Eqs. (109.32) - (??) can be reduced to the following matrix notations (Haskell, 1953):

\[
S^{(m-1)} = E_m[\Delta''_m + \Delta'_m, \Delta''_m - \Delta'_m, \omega''_m - \omega'_m, \omega''_m + \omega'_m]^T
\]

(109.36)

\[
S^{(m)} = D_m[\Delta''_m + \Delta'_m, \Delta''_m - \Delta'_m, \omega''_m - \omega'_m, \omega''_m + \omega'_m]^T
\]

(109.37)

where transformation matrix \( E_m \) and \( D_m \) are given in Appendix (Eqs. ?? and ??). The recurrence relation between \( S^{(m)} \) and \( S^{(m-1)} \) then can be established as Eq. 109.38, where \( G_m = D_mE_m^{-1} \).

\[
S^{(m)} = D_mE_m^{-1}S^{(m-1)} = G_mS^{(m-1)}
\]

(109.38)

Recursively applying Eq. 109.38 leads to Eq. 109.39. Using the relation between \( S^{(m-1)} \) and \( \Delta_m, \omega_m \), Eq. 109.40 bridges the gap between the upper boundary (i.e., response at ground surface \( S^{(0)} \)) and lower boundary (i.e., solutions of wave incident layer \( \Delta_n \) and \( \omega_n \)), upon which specific boundary conditions can be imposed.

\[
S^{(n-1)} = \prod_{i=1}^{n-1} G_iS^{(0)}
\]

(109.39)

\[
S^{(0)} = L[\Delta''_n + \Delta'_n, \Delta''_n - \Delta'_n, \omega''_n - \omega'_n, \omega''_n + \omega'_n]^T
\]

\[
L = (\prod_{i=1}^{n-1} G_i)^{-1}E_n
\]

(109.40)

Following boundary conditions are incorporated: (1) At \( n \)th layer, the incident in-plane \( P \) and \( SV \) wave potential magnitude \( \phi'_n \) and \( \Psi'_n \) are given as \( K_1 \) and \( K_2 \); (2) At the ground surface \((z = 0)\), the traction is free (i.e., the third and fourth component of \( S^{(0)} \) is 0). Therefore, the reflected dilatational wave magnitude and rotational wave magnitude can be solved by Eq. 109.41, where \( \Delta'_n = -K_1\omega^2/\alpha''_n \) and \( \omega'_n = K_2\omega^2/(2\beta''_n) \).

\[
\begin{bmatrix}
\Delta''_n \\
\omega''_n
\end{bmatrix} = \begin{bmatrix}
L_{31} + L_{32} & L_{33} + L_{34} \\
L_{41} + L_{42} & L_{43} + L_{44}
\end{bmatrix}^{-1} \begin{bmatrix}
(L_{32} - L_{31})\Delta'_n + (L_{33} - L_{34})\omega'_n \\
(L_{42} - L_{41})\Delta'_n + (L_{43} - L_{44})\omega'_n
\end{bmatrix}
\]

(109.41)
Finally, recurrence relation Eq. 109.42 can be used to trace back dilatational wave magnitude $\Delta_m$ and rotational wave magnitudes $\omega_m$ for the rest $n-1$ layers. Based on these solved dilatational and rotational magnitudes of each layer, the whole displacement and stress field can be easily computed following Eqs. (109.32)-(109.42).

\[
\begin{bmatrix}
\Delta''_{m-1} + \Delta'_{m-1} \\
\Delta''_{m-1} - \Delta'_{m-1} \\
\omega''_{m-1} - \omega'_{m-1} \\
\omega''_{m-1} + \omega'_{m-1}
\end{bmatrix} = D_{m-1}^{-1} E_m
\begin{bmatrix}
\Delta''_m + \Delta'_m \\
\Delta''_m - \Delta'_m \\
\omega''_m - \omega'_m \\
\omega''_m + \omega'_m
\end{bmatrix}
\] (109.42)

In addition, viscosity can also be included with slight modification. Considering Kelvin-Voigt viscoelastic material (Chirita et al., 2008), viscosity can be handled with complex Lame modulus and wave velocities as shown in Eq. 109.43, where $\xi$ is the damping ratio.

\[
G^* = G(1 + 2\xi i)\quad \beta^*_m \simeq \beta_m(1 + \xi i)\quad \alpha^*_m \simeq \alpha_m(1 + \xi i)
\] (109.43)

109.2.6 Seismic Motions: Large Scale Geophysical Models

109.2.6.1 Regional Seismic Motion Modeling using Serpentine Wave Propagation, SW4
109.2.7 Site Response

Site response is ...

This is part of free field motions section and just uses free field motions for producing site response.

109.2.8 Seismic Motion Incoherence

Seismic motion incoherence (as it is called for frequency domain analysis, for time domain analysis it is called lack of correlation) is a phenomena that results in spatial variability of ground motions over small distances. Significant work has been done in researching seismic motion incoherence over the last few decades. The main sources of lack of spatial correlation, according to Zerva (2009) are due to:

- Attenuation effects,
- Wave passage effects,
- Scattering effects,
- Extended source effects

Figure 109.4 shows an illustration of main sources of lack of correlation.

Figure 109.4: Four main sources contributing to the lack of correlation of seismic waves as measured at two observation points.

1. **Attenuation effects** are responsible for change in amplitude and phase of seismic motions due to the distance between observation points and losses (damping, energy dissipation) that seismic wave experiences along that distance. This is a significant source of lack of correlation for long structures (bridges), however for NPPSSS it is not of much significance.

2. **Wave passage effects** contribute to lack of correlation due to difference in recorded wave field at two location points as the (surface) wave travels, propagates from first to second point.
3. **Scattering effects** are responsible to lack of correlation by creating a scattered wave field. Scattering is due to (unknown or not known enough) subsurface geologic features that contribute to modification of wave field.

4. **Extended source effects** contribute to lack of correlation by creating a complex wave source field, as the fault ruptures, rupture propagates and generate seismic sources along the fault. Seismic energy is thus emitted from different points (along the rupturing fault) and will have different travel path and timing as it makes it observation points.

Early studies concluded that the correlation of motions increases as the separation distance between observation points decreases. In addition to that, correlation increased for decrease in frequency of observed motions. Moreover, there is a strong probabilistic nature of this phenomena, as significant uncertainty is present in relation to all four sources of lack of correlation, mentioned above. A number of excellent references are available on the subject of incoherent (or lacking correlation) seismic motions Abrahamson et al. (1991); Roblee et al. (1996); Abrahamson (1992a, 2005, 1992b); Zerva and Zervas (2002); Liao and Zerva (2006); Zerva (2009)

It is very important to note that all current models for modeling incoherent seismic motions make an ergodic assumption. That is all the models assume that a variability of seismic motions at a single site – source combination will be the same as variability in the ground motions from a data set that was collected over different site and source locations Walling (2009). Unfortunately, there does not exist a large enough data set for east North American seismic seismic events that can be used to develop incoherence models. Rather, there are models that are used to model possible incoherent behavior for east North American seismic wave propagations.

### 109.2.8.1 Lack of Correlation Modeling and Simulation

Incorporation of lack of correlation effects in seismic motions can be done using the following methods:

- along 1D, in one direction, usually one of the horizontal directions, where all the points that are the same distance (in 1D) from a control point, plane) share a single lack of correlation,

- along 2D, in two directions, usually in a vertical plane or in a horizontal plane, where all the points in a set of vertical planes or a set of horizontal planes, share same lack of correlation, and

- in full 3C, where every point of interest (in a 3D volume of soil/rock) has it own, specified, lack of correlation.
The method used here is using the so called seed motions, motions obtained through DRM in full 3C (inclined, body and surface waves), that are then enriched with appropriate uncorrelated (incoherent) components. It is important to note that only translational motions are used to model incoherence effects, while the rotational motions are not perturbed/made incoherent. This stems from the fact that currently vast majority of seismic recording stations only record translational motions and that only those translational motions are used to develop incoherent motions models. Code developed by Abrahamson (1992b) is used for this purpose.

109.2.9 Lack of Volume Change Data for Soil

Use of $G/G_{max}$ and damping curves for describing and calibrating material behavior of soil is missing a very important (crucial) information about soil/rock volume change during shearing deformation. Volume change data is very important for soil behavior. It is important to emphasize that soil behavior is very much a function of volumetric response during shear. During shearing of soil there are two essential types of soil behavior:

- Dilative (dense) soils will increase volume due to shearing
- Compressive (loose) soils will decrease volume due to shearing

The soil volume response, that is not provided by $G/G_{max}$ and damping curves data can significantly affect response due to volume constraints of soil. For example, for one dimensional site response (1C wave propagation, vertically propagating (SV) shear waves) the soil will try to change its volume (dilate if it is dense or compress if it is loose). However, such volume change can only happen vertically (since there is no constraint (foundation for example) on top, while horizontally the soil will be constraint by other soil. That means that any intended volume change in horizontal direction will be resisted by change in (increase for dense and decrease for loose soil) horizontal stress. For example for dilative (dense) soil, additional horizontal stress will contribute to the increase in mean pressure (confinement) of the soil, thus increasing the stiffness of that soil. It is the opposite for compressive soil where shearing will result in a reduction of confinement stress. Figure 109.5 shows three responses for no-volume change (left), compressive (middle) and dilative (right) soil with full volume constraint, resulting in changes in stiffness for compressive (reduction in stiffness), and dilative (increase in stiffness).

Changes in stiffness of soil during shearing deformation will influence wave propagation and amplification of different frequencies. Figure 109.6 shows response of no-volume change soil (as it is/should be assumed if only $G/G_{max}$ and damping curves are available, with no volume change data) and a response of a dilative soil which stiffens up during shaking due to restricted intent to dilate. It is clear that dilative soil will show significant amplification of higher frequencies.
Figure 109.5: Constitutive Cyclic response of soils with constraint volumetric deformation: (left) no volume change (soil is at the critical state); (middle) compressive response with decrease in stiffness; (right) dilative response with increase in stiffness.

Figure 109.6: One dimensional seismic wave propagation through no-volume change and dilative soil. Please note the (significant) increase in frequency of motions for dilative soil. Left plot is a time history of motions, while the right plot shows amplitudes at different frequencies.
109.3 Earthquake Soil Structure Interaction

Current design practice for structures subject to earthquake loading regards dynamic SSII to be mainly beneficial to the behavior of structures (Jeremić and Preisig, 2005). Including the flexibility of the foundation reduces the overall stiffness of a system and therefore reduces peak loads caused by a given ground motion. Even if this is true in most cases there is the possibility of resonance occurring as a result of a shift of the natural frequencies of the SSI-system. This can lead to large inertial forces acting on a structure.

As a result of these large inertial forces caused by the structure oscillating in its natural frequency the structure as well as the soil surrounding the foundation can undergo plastic deformations. This in turn further modifies the overall stiffness of the SSI-system and makes any prediction on the behavior very difficult.

Dynamic SSII also becomes important in the design of large infrastructure projects. As authorities and insurance companies try to introduce the concept of performance based design to the engineering community more sophisticated models are needed in order to obtain the engineering demand parameters (EDP’s). A good numerical model of a soil-foundation-structure system can therefore not only prevent the collapse or damage of a structure but also help to save money by optimizing the design to withstand an earthquake with a certain return period.

A variety of methods of different levels of complexity are currently being used by engineers. In the following an overview over the most important ones is presented. A more thorough discussion on methods and specific aspects of dynamic SSII is available in Wolf (1985) and more recently in Wolf and Song (2002).

- **No SSII**

  The ground motion is applied directly to the base of the building. Alternatively, instead of applying the ground motion directly to the base of the structure, effective earthquake forces proportional to the base acceleration can be applied to the nodes.

  This procedure is reasonable only for flexible structures on very stiff soil or rock. In this case the displacement of the ground doesn’t get modified by the presence of the structure. For stiffer structures on soil the ground motion has to be applied to the soil. The model has to incorporate propagation of the motion through the soil, its interaction with the structure and the radiation away from the structure.

- **Direct methods**

  Direct methods treat the SSI-system as a whole. The numerical model incorporates the spatial
discretization of the structure and the soil. The analysis of the entire system is carried out in one step. This method provides most generality as it is capable of incorporating all nonlinear behavior of the structure, the soil and also the interface between those two (sliding, uplift).

- **Substructure methods**

Substructure methods refer to the principle of superposition. The SSI-system is generally subdivided into a structure part and a soil part. Both substructures can be analyzed separately and the total displacement can be obtained by adding the contributions at the nodes on the interface.

This method reduces the size of the problem considerably. As the time needed for an analysis doesn’t increase linearly with an increasing number of equations the substructure method is much faster than the direct method. The biggest drawback of the method however is the fact that linearity is assumed. For nonlinear systems the substructure method cannot be used.

For the direct method different levels of sophistication are possible:

- **Foundation stiffness approach**

  The behavior of the soil is accounted for by simple mechanical elements such as springs, masses and dash pots. Different configurations of the subsoil can be taken into account by connecting several springs, masses and dash pots whose parameters have been determined by a curve fitting procedure Wolf (1994). This approach is very popular among structural engineers as it is relatively easy to be integrated in a commonly used finite element code.

  Other methods use frequency dependent springs and dash pots and therefore require an analysis in frequency domain. Relatively complex configurations of layered subsoil and embedded foundations can be modeled with good accuracy by replacing the (elastic) soil with a sequence of conical rods Wolf and Song (2002) and Wolf and Preisig (2003).

- **p-y methods**

  Attempts have been made to apply the static p-y approach for evaluating lateral loading on pile foundations to dynamic problems. Mostafa and El Naggar (2002) lists several references and provides a parametric study of single piles and pile groups in different soil types under simplified loading cases.

  Even if p-y curves are widely used for estimating lateral loading on piles they are rarely used in full dynamic soil-structure interaction analysis. Current work trying to implement these methods into finite element codes is likely to make them more popular with the engineering community.
• Full 3d

Full nonlinear three-dimensional modeling of dynamic soil-foundation-structure interaction can be regarded as the 'brute force' approach. Displacements and forces can be obtained not only for the structure as in the above mentioned methods but also for the soil. In spite of the computational resources and modeling effort required for an analysis it is the only method that remains valid for all kinds of problems involving material nonlinearities, contact/interface problems, different loading cases and complex geometries.

109.4 Earthquake Soil Structure Interaction Modeling Details

109.4.1 Seismic Motions Input into Finite Element Model

A number of methods is used to input seismic motions into finite element model. Most of them are based on simple intuitive approaches, and as such are not based on rational mechanics. Most of those currently still widely used methods cannot properly model all three components of body waves as well as always present surface waves. There exist a method that is based on rational mechanics and can model both body and surface seismic waves input into finite element models with high accuracy. That method is called the Domain Reduction Method (DRM) (Bielak et al., 2003a; Yoshimura et al., 2003a). The DRM aims to reduce the large computational domain, encompassing fault, rock, soil and the structure, to a much smaller domain, encompassing only local soil and the structure. The method was developed with earthquake ground motions in mind, with the main idea to replace the force couples at the fault with their counterpart acting on a continuous surface surrounding local feature of interest. The local feature can be any geologic or man made object that constitutes a difference from the simplified large domain, free field, for which displacements and accelerations are easier to obtain.

The DRM is applicable to a much wider range of problems. It is essentially a variant of global–local set of methods and as formulated can be used for any problems where the local feature can be bounded by a continuous surface (that can be closed or not). The local feature in general can represent a soil–foundation–structure system (bridge, building, dam, tunnel...), or it can be a crack in large domain, or some other type of inhomogeneity that is fairly small compared to the size of domain where it is found.

In what follow, the DRM is developed in a somewhat different way than it was done in original papers by Bielak et al. (2003a); Yoshimura et al. (2003a)). The main features of the DRM are then analyzed and appropriate practical modeling issues addressed.
109.4.1.1 The Domain Reduction Method (DRM) Development

A large physical domain is to be analyzed for dynamic behavior. The source of disturbance is a known time history of a force field $P_e(t)$. That source of loading is far away from a local feature which is dynamically excited by $P_e(t)$ (see Figure 109.7).

![Diagram of large physical domain with source of load $P_e(t)$ and local feature](image)

Figure 109.7: Large physical domain with the source of load $P_e(t)$ and the local feature (in this case a soil-structure system).

The system to be analyzed can be quite large, for example earthquake hypocenter can be many kilometers away from the local feature of interest. Similarly, the small local feature in a machine part can be many centimeters away from the source of dynamic loading which influences this local feature. In this sense the term large domain is relative to the size of the local feature and the distance to the dynamic forcing source.

It would be beneficial not to analyze the complete system, as we are only interested in the behavior of the local feature and its immediate surrounding, and can almost neglect the domain outside of some relatively close boundaries. In order to do this, we need to somehow transfer the loading from the source to the immediate vicinity of the local feature. For example we can try to reduce the size of the domain to a much smaller model bounded by surface $\Gamma$ as shown in Figure 109.7. In doing so we must ensure that the dynamic forces $P_e(t)$ are appropriately propagated to the much smaller model boundaries $\Gamma$.

**DRM Formulation** In order to appropriately propagate dynamic forces $P_e(t)$ one actually has to solve the large scale problem which will include the effects of the local feature. Most of the time this is impossible as it involves all the complexities of large scale computations and relatively small local feature. Besides, the main goal of presented developments is to somehow reduce the large scale domain
as to be able to analyze in details behavior of the local feature.

In order to propagate consistently the dynamic forces \( P_e(t) \) we will make a simplification in that we will replace a local feature with a simpler domain that is much easier to be analyzed. That is, we replace the local feature (bridge, building, tunnel, crack) with a much simpler geometry and material. For example, Figure 109.8 shows a simplified model, without a foundation–building system. The idea is to simplify the model so that it is much easier to consistently propagate the dynamic forces to the boundary \( \Gamma \). The notion that it is much easier to propagate those dynamic forces is of course relative. This is still a very complex problem, but at least the influence of local feature is temporarily taken out.

It is convenient to name different parts of domain. For example, the domain inside the boundary \( \Gamma \) is named \( \Omega_0 \). The rest of the large scale domain, outside boundary \( \Gamma \), is then named \( \Omega^+ \). The outside domain \( \Omega^+ \) is still the same as in the original model, while the change, simplification, is done on the domain inside boundary \( \Gamma \). The displacement fields for exterior, boundary and interior of the boundary \( \Gamma \) are \( u_e \), \( u_b \) and \( u_i \), respectively, on the original domain.

The equations of motions for the complete system can be written as

\[
\begin{bmatrix}
M \\
K
\end{bmatrix}
\begin{bmatrix}
\ddot{u} \\
u
\end{bmatrix}
= \begin{bmatrix}
P_e
\end{bmatrix}
\tag{109.44}
\]

or, if written for each domain (interior, boundary and exterior of \( \Gamma \)) separately, the equations obtain the

\[
\begin{bmatrix}
M_0 \\
K_0
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_0 \\
u_0
\end{bmatrix}
= \begin{bmatrix}
P_{e0}
\end{bmatrix}
\]

\[
\begin{bmatrix}
M_\Gamma \\
K_\Gamma
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_\Gamma \\
u_\Gamma
\end{bmatrix}
= \begin{bmatrix}
P_{e\Gamma}
\end{bmatrix}
\]

\[
\begin{bmatrix}
M_+ \\
K_+
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_+ \\
u_+
\end{bmatrix}
= \begin{bmatrix}
P_{e+}
\end{bmatrix}
\]
following form:

\[
\begin{bmatrix}
M^{\Omega}_{ii} & M^{\Omega}_{ib} \\
M^{\Omega}_{bi} & M^{\Omega}_{bb} + M^{\Omega+}_{bb} + M^{\Omega+}_{eb}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b
\end{bmatrix}
+ \begin{bmatrix}
K^{\Omega}_{ii} & K^{\Omega}_{ib} \\
K^{\Omega}_{bi} & K^{\Omega}_{bb} + K^{\Omega+}_{bb} + K^{\Omega+}_{eb}
\end{bmatrix}
\begin{bmatrix}
u_i \\
u_b
\end{bmatrix}
= \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

(109.45)

In these equations, the matrices \( M \) and \( K \) are mass and stiffness matrices respectively; the subscripts \( i, e, \) and \( b \) are referencing nodes in either the interior \((i)\) or exterior \((e)\) domain or on their common boundary \((b)\), while the superscripts \( \Omega \) and \( \Omega^+ \) reference domains to which matrices belong.

The previous equation can be separated provided that we maintain the compatibility of displacements and equilibrium. The resulting two equations of motion are

\[
\begin{bmatrix}
M^{\Omega}_{ii} & M^{\Omega}_{ib} \\
M^{\Omega}_{bi} & M^{\Omega}_{bb}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b
\end{bmatrix}
+ \begin{bmatrix}
K^{\Omega}_{ii} & K^{\Omega}_{ib} \\
K^{\Omega}_{bi} & K^{\Omega}_{bb}
\end{bmatrix}
\begin{bmatrix}
u_i \\
u_b
\end{bmatrix}
= \begin{bmatrix} 0 \\ P_b \end{bmatrix}, \quad \text{in} \Omega
\]

(109.46)

and

\[
\begin{bmatrix}
M^{\Omega+}_{bb} & M^{\Omega+}_{be} \\
M^{\Omega+}_{eb} & M^{\Omega+}_{ee}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_b \\
\ddot{u}_e
\end{bmatrix}
+ \begin{bmatrix}
K^{\Omega+}_{bb} & K^{\Omega+}_{be} \\
K^{\Omega+}_{eb} & K^{\Omega+}_{ee}
\end{bmatrix}
\begin{bmatrix}
u_b \\
u_e
\end{bmatrix}
= \begin{bmatrix} -P_b \\ P_e \end{bmatrix}, \quad \text{in} \Omega^+
\]

(109.47)

Compatibility of displacements is maintained automatically since both equations contain boundary displacements \( u_b \) (on boundary \( \Gamma \)), while the equilibrium is maintained through action–reaction forces \( P_b \).

In order to simplify the problem, the local feature is removed from the interior domain. Thus, the interior domain is significantly simplified. In other words, the exterior region and the material therein are identical to those of the original problem as the dynamic force source. On the other hand, the interior domain (denoted as \( \Omega_0 \)), is simplified, the localized features is removed (as seen in figure 109.8).

For this simplified model, the displacement field (interior, boundary and exterior, respectively) and action–reaction forces are denoted by \( u^0_i, u^0_b, u^0_e \) and \( P^0_b \). The entire simplified domain \( \Omega_0 \) and \( \Omega^+ \) is now easier to analyze.

The equations of motion in \( \Omega^+ \) for the auxiliary problem can now be written as:

\[
\begin{bmatrix}
M^{\Omega^+}_{bb} & M^{\Omega^+}_{be} \\
M^{\Omega^+}_{eb} & M^{\Omega^+}_{ee}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_b^0 \\
\ddot{u}_e^0
\end{bmatrix}
+ \begin{bmatrix}
K^{\Omega^+}_{bb} & K^{\Omega^+}_{be} \\
K^{\Omega^+}_{eb} & K^{\Omega^+}_{ee}
\end{bmatrix}
\begin{bmatrix}
u_b^0 \\
u_e^0
\end{bmatrix}
= \begin{bmatrix} -P^0_b \\ P_e \end{bmatrix}
\]

(109.48)

Since there was no change to the exterior domain \( \Omega^+ \) (material, geometry and the dynamic source are still the same) the mass and stiffness matrices and the nodal force \( P_e \) are the same as in Equations (109.46) and (109.47).

Previous equation 109.48 can be used to obtain the dynamic force \( P_e \):

\[
P_e = M^{\Omega^+}_{eb} \ddot{u}_b^0 + M^{\Omega^+}_{ee} \ddot{u}_e^0 + K^{\Omega^+}_{eb} u_b^0 + K^{\Omega^+}_{ee} u_e^0
\]

(109.49)
The total displacement, $u_e$, can be expressed as the sum of the free field $u_e^0$ (from the background, simplified model) and the residual field $w_e$ (coming from the local feature) as following:

$$u_e = u_e^0 + w_e \quad (109.50)$$

It is important to note that this is just a change of variables and not an application of the principle of superposition. The residual displacement field, $w_e$ is measured relative to the reference free field $u_e^0$.

By substituting Equation 109.50 in Equation 109.45 one obtains:

$$\begin{bmatrix}
M_{ii}^\Omega & M_{ib}^\Omega & 0 \\
M_{bi}^\Omega & M_{bb}^\Omega & M_{be}^\Omega \\
0 & M_{eb}^\Omega & M_{ee}^\Omega
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b \\
\ddot{w}_e
\end{bmatrix}
+ \begin{bmatrix}
K_{ii}^\Omega & K_{ib}^\Omega & 0 \\
K_{bi}^\Omega & K_{bb}^\Omega + K_{be}^\Omega & K_{be}^\Omega \\
0 & K_{eb}^\Omega & K_{ee}^\Omega
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b \\
\ddot{w}_e + \ddot{w}_e
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
P_e
\end{bmatrix} \quad (109.51)$$

which, after moving the free field motions $u_e^0$ to the right hand side, becomes

$$\begin{bmatrix}
M_{ii}^\Omega & M_{ib}^\Omega & 0 \\
M_{bi}^\Omega & M_{bb}^\Omega + M_{be}^\Omega & M_{be}^\Omega \\
0 & M_{eb}^\Omega & M_{ee}^\Omega
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b \\
\ddot{w}_e
\end{bmatrix}
+ \begin{bmatrix}
K_{ii}^\Omega & K_{ib}^\Omega & 0 \\
K_{bi}^\Omega & K_{bb}^\Omega + K_{be}^\Omega & K_{be}^\Omega \\
0 & K_{eb}^\Omega & K_{ee}^\Omega
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b \\
\ddot{w}_e
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix} \quad (109.52)$$

By substituting Equation 109.49 in previous Equation 109.52, the right hand side can now be written as

$$\begin{bmatrix}
M_{ii}^\Omega & M_{ib}^\Omega & 0 \\
M_{bi}^\Omega & M_{bb}^\Omega + M_{be}^\Omega & M_{be}^\Omega \\
0 & M_{eb}^\Omega & M_{ee}^\Omega
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b \\
\ddot{w}_e
\end{bmatrix}
+ \begin{bmatrix}
K_{ii}^\Omega & K_{ib}^\Omega & 0 \\
K_{bi}^\Omega & K_{bb}^\Omega + K_{be}^\Omega & K_{be}^\Omega \\
0 & K_{eb}^\Omega & K_{ee}^\Omega
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_i \\
\ddot{u}_b \\
\ddot{w}_e
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix} \quad (109.53)$$

The right hand side of equation 109.53 is a dynamically consistent replacement force, the so called effective force, $P_{eff}$ for the dynamic source forces $P_e$. In other words, the dynamic force $P_e$ was
consistently replaced by the effective force $P_{\text{eff}}$:

$$P_{\text{eff}} = \begin{cases} P_{\text{eff}}^i \\ P_{\text{eff}}^b \\ P_{\text{eff}}^e \end{cases} = \begin{cases} 0 \\ -M_{\Omega^+} \ddot{u}_e^0 - K_{\Omega^+} \dot{u}_e^0 \\ M_{\Omega^+} \ddot{u}_b^0 + K_{\Omega^+} \dot{u}_b^0 \end{cases}$$

(109.54)

**DRM Discussion**

**Single Layer of Elements used for $P_{\text{eff}}$.** The Equation (109.54) shows that the effective nodal forces $P_{\text{eff}}$ involve only the sub-matrices $M_{\Omega^+}, K_{\Omega^+}, M_{\Omega^+}, K_{\Omega^+}$. These matrices vanish everywhere except the single layer of finite elements in domain $\Omega^+$ adjacent to $\Gamma$. The significance of this is that the only wave-field (displacements and accelerations) needed to determine effective forces $P_{\text{eff}}$ is that obtained from the simplified (auxiliary) problem at the nodes that lie on and between boundaries $\Gamma$ and $\Gamma_e$, as shown in Figure 109.9.

![Figure 109.9](image)

Figure 109.9: DRM: Single layer of elements between $\Gamma$ and $\Gamma_e$ is used to create $P_{\text{eff}}$, for a section of 8 node brick.

Figure 109.10 show boundary and external nodes for a section of a 27 node brick. Please note that for a 20, 20-17 and 27 node bricks, boundary nodes are nodes that belong to $\Gamma$ surface, so for 27 node brick there are 9 nodes on that face, while the external nodes are all the nodes that are not boundary nodes, there will be 18 of those nodes.

**Only residual waves outgoing.** Another interesting observation is that the solution to problem described in Equation (109.53) comprises full unknowns (displacements and accelerations) inside and on the boundary $\Gamma$ ($u_i$ and $u_b$ respectively). On the other hand, the solution for the domain outside single layer of finite elements (outside $\Gamma_e$) is obtained for the residual unknown (displacement and accelerations)
field, $w_e$ only. This residual unknown field is measured relative to the reference free field of unknowns (see comments on page 541). That effectively means that the solution to the equation Equation (109.53) outside the boundary $\Gamma_e$ will only contain additional waves field resulting from the presence of a local feature. This in turn means that if the interest is in behavior of local feature and the surrounding media (all within boundary $\Gamma$) one can neglect the behavior of the full model (outside $\Gamma_e$ in $\Omega^+$) and provide appropriate supports (including fixity and damping) at some distance from the boundary $\Gamma_e$ into region $\Omega^+$. This is significant for a number of reasons:

- large models can be reduced in size to encompass just a few layers of elements outside boundary $\Gamma_e$ (significant reduction for, say earthquake problems where the size of a local feature is orders of magnitudes smaller then the distance to the dynamic source force $P_e$ (earthquake hypocenter).

- the residual unknown field can be monitored and analyzed for information about the dynamic characteristics of the local feature. Since the residual wave field is $w_e$ is measured relative to the reference free field $u^0_e$, the solution for $w_e$ has all the characteristics of the additional wave field stemming from the local feature.

**Inside domain $\Omega$ can be inelastic.** In all the derivations in section 109.4.1.1 no restriction was made on the type of material inside the plastic bowl (inside $\Gamma_e$). That is, the assumption that the material inside is linear elastic is not necessary as the DRM is not relying on principle of superposition. The Equation 109.50 was only describing the change of variables, and clearly there was no use of the principle of superposition, which is only valid for linear elastic solids and structures. It is therefore possible to assume that the derivations will still be valid with any type of material (linear or nonlinear, elastic.
or inelastic) inside $\Gamma_e$. With this in mind, the DRM becomes a very powerful method for analysis of soil–foundation–structure systems.

**All types of realistic seismic waves are modeled.** Since the effective forcing $P_{eff}$ consistently replaces the effects of the seismic source, all appropriate (real) seismic waves are properly (analytically) modeled, including body (SV, SH, P) and surface (Rayleigh, Love, etc...) waves.

**Properties of finite elements inside the DRM Layer.** The DRM layers, a single layer of finite elements just outside $\Gamma$ surfaces, where effective DRM forces $P_{eff}$ are applied, needs to be carefully modeled. A number of conditions regarding the DRM layer need to be taken into account:

- The finite elements within the DRM layer need to be linear elastic.
- Material models for the finite elements within the DRM layer need to have same, or very similar material properties as the elastic part of material properties as the material inside the DRM layer. Although material inside the DRM layer can be elastic-plastic, it is beneficial if the linear elastic portion of material properties, for example for nonlinear elastic material at zero strain or for elastic-plastic material, elastic properties inside yield surface, for the DRM layer, is same, similar to the material used inside the DRM layer. All of the used elastic material properties need to be same as elastic material properties used for free field analysis in order to have consistent wave field.
- Dimensions of the DRM layer finite elements (thickness of the DRM layer) need to follow the same rule for element size (depending on chosen stiffness) so that there is no artificial (mesh dependent) filtering above certain frequencies. That means that 10 linear interpolation finite elements (8 node bricks) or 2 quadratic interpolation elements (27 node bricks) are needed per wave length (Bathe and Wilson, 1976; Hughes, 1987; Argyris and Mlejnek, 1991). For example if maximum modeling frequency is $f_{max} = 20$ Hz, and wave length is given as $\lambda_{min} = v / f_{max}$, where $v$ is the wave velocity, maximum grid spacing (element size) for linear interpolation elements $\Delta h^{LE}$ should not exceed

$$\Delta h^{LE} \leq \frac{\lambda}{10} = \frac{v}{10 f_{max}} = \frac{v}{10 \times 20 \text{ Hz}} = \frac{v}{200 \text{ Hz}}$$

while for quadratic interpolation elements such grid spacing (element size) $\Delta h^{QE}$ is limited to

$$\Delta h^{QE} \leq \frac{\lambda}{2} = \frac{v}{2 f_{max}} = \frac{v}{2 \times 20 \text{ Hz}} = \frac{v}{40 \text{ Hz}}$$

The wave velocity $v$ is the lowest wave velocity that is of interest in the simulation, usually the shear wave velocity.
Properties of the finite elements outside of the DRM Layer. Finite elements outside DRM layer need special considerations as well.

- Material outside the DRM layer needs to be linear elastic with addition of viscous damping, Caughey, Rayleigh, etc.

- At least one layer of finite elements outside DRM layer needs to be provided. If damping of outgoing waves, the so called radiation damping, is to be modeled, then more than one layer of finite elements outside DRM needs to be provided.

- If radiation damping is modeled, it is recommended to have more than just two layers, outside DRM layer. For example 4 or 5 additional layers work quite well.

- First layer outside DRM layer needs to be linear elastic, of similar/same properties as material inside DRM layer, and with NO viscous damping. The reason for this requirement, is explained by the fact that $P_{eff}$ force, see equation 109.54 on page 542, is applied to finite elements within DRM layer. If finite elements that are outside/adjacent to the DRM layer have large viscous damping, then $P_{eff}$ forces will be producing potentially significant reaction forces from large viscous damping that is placed on nodes of elements that are shared with DRM finite element nodes, and nodes of finite elements just outside DRM layer, and are connected to DRM finite elements. These reaction, viscous forces, will affects, change $P_{eff}$ forces in a way that will not be consistent with seismic wave field that was used to develop $P_{eff}$.

- Viscous damping, Caughey, Rayleigh, should be placed on finite elements outside this first layer of elements, that is outside, adjacent to DRM layer, in order to damp out outgoing waves, the differential wave field, the ”$w_e$“ waves, see equation 109.50 on page 541. Additional viscous damping layers are added to damp out any additional waves, the so called radiation damping from structural oscilations. Damping in those additional layers is to be progressively larger, much larger than physical viscous damping. Values of equivalent damping of 20%, then 30%, then 50%, or higher have been used in order to damp outgoing waves, to model radiation damping.

A Note on Input Motions for DRM. Seismic motions (free field) that are used for input into a DRM model need to be consistent. In other words, a free field seismic wave that is used needs to fully satisfy equations of motion. For example, if free field motions are developed using a tool (SHAKE, or EDT or SW4, or fk, &c.) using time step $\Delta t = 0.01s$ and then you decide that you want to run your analysis with a time step of $\Delta t = 0.001s$, simple interpolation (10 additional steps for each of the original steps) might create problems. Simple linear interpolation actually might (will) not satisfy wave propagation
equations and if used will introduce additional, high frequency motions into the model. It is a very good idea to generate free field motions with the same time step as it will be used in ESSI simulation.

Similar problem might occur if spacial interpolation is done, that is if location of free field model nodes is not very close to the actual DRM nodes used in ESSI model. Spatial interpolation problems are actually a bit less acute, however one still has to pay attention and test the ESSI model for free conditions and only then add the structure(s) on top.

Input motions for the DRM are based on Free Field motions, that can be developed by a number of methods, as described in section 502.2.3 on page 2205.
Chapter 110

Parallel Computing in Computational Mechanics


(In collaboration with Dr. Guanzhou Jie and Dr. Yuan Feng, and Dr. Han Yang)
110.1 Chapter Summary and Highlights

110.2 Introduction

110.2.1 High Performance Computing on DMPs, SMPs, GPGPUs, FPGA

110.2.1.1 Distributed Memory Parallel (SMP) Computations

110.2.1.2 Shared Memory Parallel (SMP) Computations

110.2.1.3 General Purpose Graphical Processing Units (GPGPUs)

110.2.1.4 Fast Programmable Gate Arrays (FPGAs)

110.2.2 Parallel Computing for Elastic-Plastic Solids and Structures

110.2.3 Problem Requirements

Stages, Increments, Iterations

For many classes of scientific simulations, an initial (static) decomposition of a finite element mesh needs to be computed such that the number of mesh elements assigned to each processor is roughly equal and the number of mesh elements that are adjacent to elements assigned to other processors (i.e., the size of the subdomain boundary) is minimized. Ensuring that the number of mesh elements is balanced will result in a load-balanced computation, while minimizing the size of the subdomain boundary will minimize the inter-processor communications overhead. Such a decomposition is usually obtained by a graph partitioning algorithm. Recently, a number of multilevel graph partitioning algorithms (e.g. Hendrickson and Leland (1995), Karypis and Kumar (1998b,a), Monien et al. (1999), Walshaw and Cross (1999)) have been developed that are able to compute excellent static decompositions for a wide range of scientific simulations.

110.2.3.1 Finite Element Computations in Geomechanics

The distinct feature of elasto–plastic finite element computations is the presence of two iteration levels. In a standard displacement based finite element implementation, constitutive driver at each Gauss point iterates in stress and internal variable space, computes the updated stress state, constitutive stiffness tensor and delivers them to the finite element functions. Finite element functions then use the updated stresses and stiffness tensors to integrate new (internal) nodal forces and element stiffness matrix. Then, on global level, nonlinear equations are iterated on until equilibrium between internal and external forces is satisfied within some tolerance. In more details:


Elastic computations. In the case of elastic computations constitutive driver has a simple task of computing increment in stresses ($\Delta \sigma_{ij}$) for a given deformation increment ($\Delta \epsilon_{kl}$), through a closed form equation ($\Delta \sigma_{ij} = E_{ijkl} \Delta \epsilon_{kl}$) (Jeremić and Sture, 1997). It is important to note that in this case the amount of work per Gauss point is known in advance. The amount of computational work is the same for every integration point. If we assume the same number of integration points per element, it follows that the amount of computational work is the same for each element and it is known in advance.

Elasto–plastic computations. On the other hand, for elasto–plastic problems, for a given incremental deformation the constitutive driver is iterating in stress and internal variable space until consistency condition is satisfied ($F = 0$). The number of iterations is not known in advance. Initially, all Gauss points are in elastic range, but as we incrementally apply loads, the plastic zones develop. For Gauss points still in elastic range, there are no iterations, the constitutive driver just computes incremental stresses from closed form solution. Computational load will increase significantly for integration of constitutive equations in plastic range. In particular, constitutive level integration algorithms (Jeremić et al., 1998, 1999; Jeremić and Yang, 2002) for soils are very computationally demanding. From the experience of the PI, more than 70% of wall clock time during an elasto–plastic finite element analysis is spent in constitutive level iterations. This is in sharp contrast with elastic computations where the dominant part is solving the system of equations which consumes about 80% of run time. The extent of additional, constitutive level iterations is not known before the actual computations are over. In other words, the extent of elastic-plastic domain is not known ahead of time.

The traditional preprocessing type of DD method (also known as topological DD) splits domain based on the initial geometry and assigns roughly the same number of elements to every computational node and minimizes the size of subdomain boundaries. This approach might result in serious computational load imbalance for elasto–plastic problems. For example one domain might be assigned all of the elasto–plastic elements and spend large amount of time in constitutive level iterations. The other domains will have elements in elastic state and thus spend far less computational time in computing stress increments. This results in program having to wait for the slowest domain (the one with large number of elasto–plastic finite elements) to complete constitutive level iterations and only proceed with global system iterations after that.

This illustrates a two–fold challenge with computational load balancing for elastic–plastic simulations in geomechanics.
110.2.3.2 Adaptive Computation

First, these computations are dynamic in nature. That is, the structure of elastic and elastic-plastic domains changes dynamically and unpredictably during the course of the computation. For this reason, a static decomposition computed as a pre-processing step is not sufficient to ensure the computational load-balance of the entire computation. Instead, periodic computational load-balancing is required during the course of the computation. The problem of computing a dynamic decomposition shares the same requirements as that of computing the initial decomposition (i.e., balance the mesh elements and minimize the inter-processor communications), while also requiring that the cost associated with redistributing the data in order to balance the computational load is minimized. This last requirement prevents us from simply computing a whole new static partitioning from scratch each time computational load-balancing is required.

Often, the objective of minimizing the data redistribution cost is at odds with the objective of minimizing the inter-processor communications. For applications in which the computational requirements of different regions of the domain change rapidly, or the amount of state associated with each element is relatively high, minimizing the data redistribution cost is preferred over minimizing the communications incurred during parallel processing.

For applications in which computational load-balancing occurs very infrequently, the key objective of a load-balancing algorithm is in obtaining the minimal inter-processor communications. For many application domains, it is straightforward to select a primary objective to minimize (i.e., minimize whichever cost dominates). However, one of the key issues concerning the elastic-plastic computation is that the number of iterations between computational load-balancing phases is both unpredictable and dynamic. For example, in the case of static problems, zones in the 3D solid may become plastic and then unload to elastic (during increments of loading) so that the extent of plastic zone is changing. The change can be both slow and rapid. Slow change usually occurs during initial loading phases, while the later deformation tends to localize in narrow zones rapidly and the rest of the solid unloads rapidly (becomes elastic again). The narrow, localized zone has heavy computational load on the constitutive level (in each integration point within elements). Similar phenomena is observed in seismic soil–structure interaction computations where stiff structure interacts with soft soil and elastic and elastic–plastic zones change significantly during loading cycles. In this type of computation, it is extremely difficult to select the type of computational load-balancing algorithm to employ. Furthermore, the preferred computational load-balancing algorithm is liable to change during the course of the computation, and so the selection must be made dynamically.
110.2.3.3 Multi-phase Computation

The second challenge associated with computational load-balancing elastic-plastic computations in geomechanics is that these are two-phase computations. That is, plastic computations follow up the elastic computations. There is a synchronization phase between the computations, as only after the elastic computation is finished is it possible to check if the plastic computation is required for a given integration (Gauss) point within an element. For regions of the mesh in which this check indicates that the plastic computation is necessary, lengthy plastic computations are then performed. The existence of the synchronization step between the two phases of the computation requires that each phase be individually load balanced. That is, it is not sufficient to simply sum up the relative times required for each phase and to compute a decomposition based on this sum. Doing so may lead to some processors having too much work during the elastic computation (and so, these may still be working after other processors are idle), and not enough work during the plastic computation, (and so these may be idle while other processors are still working), and vice versa. Instead, it is critical that every processor have an equal amount of work from both of the phases of the computation.

110.2.4 Parallel Computing Hardware

110.2.4.1 DMPs and SMPs

Scalability issues for SMPs Cache coherence

- Compute Nodes, CPUs, Cores,
- GPUs (band-with and latency with the main memory)
- Networks (band-with and latency) 10, 100, 1,000, 10,000, Infiniband,
- Large parallel supercomputers
- small, user owned parallel machines (clusters of clusters)

ESSI Computer

- 208 (784) CPU cores,
- 288GB (1056GB) of distributed RAM,
- 24TB (48TB) of distributed disk space, and
- dual network, InfiniBand for MPI, and GigaBit for file system

Small ESSI Computer

- 32 CPU cores (AMD), 24 CPU cores (Intel)
• 64GB RAM,
• 4TB
• on-board network

110.2.5 Parallel Computing Software

110.2.5.1 Amdahl’s Law

$n$ is a number of parallel processes

$B$ is the fraction of algorithm that is serial

Total time to finish (wall clock time) with $n$ parallel processes $T(n)$

$$T(n) = T(1)(B + \frac{1}{n}(1 - B))$$  \hspace{1cm} (110.1)

Theoretical speedup

$$S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1)(B + \frac{1}{n}(1 - B))} = \frac{1}{B + \frac{1}{n}(1 - B))}$$  \hspace{1cm} (110.2)

110.2.5.2 Static and Dynamic Graph Partitioning

google search, data mining etc.

110.2.5.3 Real parallel and embarrassingly parallel

Finite Element matrices computations (elastic and elastic-plastic)

System of equation solvers (non-iterative and iterative)

Examples for

elastic (elements (Seq) + solver(P))
elastic (elements (P) + solver (Seq))
elastic (elements (P) + solver (P))
elastic-plastic (Seq) + solver (P)
elastic-plastic (P) + solver (Seq)
elastic-plastic (P) + solver (P)
110.2.5.4 Parallel Computing for Elastic-Plastic FEM

- Current Parallel FEM are
  - Well developed for elastic FEM
  - Undeveloped for elastic–plastic FEM
  - Well developed for homogeneous distributed memory parallel (DMP) computers,
  - Undeveloped for multiple performance (multi–generation) DMPs (example MOOSE, ESSI...)

- Need: dynamic computational load balancing for
  - multiple element types,
  - multiple material models
  - multiple compute node performances
  - multiple network performance performances

110.2.5.5 Plastic Domain Decomposition

- Multi-objective optimization problem (minimize both the inter-processor communications, the data redistribution costs and create balanced partitions)

- Computational load balancing adds overhead $T_{\text{overhead}} := T_{\text{comm}} + T_{\text{regen}}$
  - $T_{\text{comm}}$ data communication load depending on network conditions.
  - $T_{\text{regen}}$ model regeneration for new partitioning, application (model) dependent

- Computational load among CPUs $T_j := \sum_{i=1}^{\text{nel}} \text{ElemCompLoad}[i], \ j = 1, ..., nCPU$

- Goal: minimize maximum compute time (slowest CPU) $T_{\text{max}} := \max(T_j) \ j = 1, ..., nCPU$

- Total compute time (not wall clock time) $T_{\text{sum}} := \sum(T_j)$

- Best execution time (perfect load balancing) $T_{\text{best}} := T_{\text{sum}}/nCPU$, $\Rightarrow T_j \equiv T_{\text{best}}$ for each $j = 1, ..., nCPU$

- Best performance gain $T_{\text{gain}} := T_{\text{max}} - T_{\text{best}}$

- Computational load balancing is beneficial iff $T_{\text{gain}} \geq T_{\text{overhead}} = T_{\text{comm}} + T_{\text{regen}}$

- Scalability (saturation, superlinear ...)

---

Jeremić et al.

University of California, Davis

110.2.5.6 Template Meta-programs

Fine grained parallelism

110.3 Plastic Domain Decomposition Algorithm

110.3.1 Introduction

Domain Decomposition approach is the most popular and effective method to implement parallel finite element method. The underlying idea is to physically divide the problem domain into subdomains and finite element calculations will be performed on each individual domain in parallel. Domain Decomposition can be overlapping or non-overlapping. The overlapping domain decomposition method divides the problem domain into several slightly overlapping subdomains. Non-overlapping domain decomposition is extensively used in continuum finite element modeling due to the relative ease to program and organize computations and is the one that will be examined in this chapter.

In general, a good non-overlapping decomposition algorithm should be able to

- handle irregular mesh of arbitrarily shaped domain.
- minimize the interface problem size by delivering minimum boundary connectivity, which will help reducing the communication overheads.

The well-known idea of domain decomposition method can be found in a 1870 paper by the father of domain decomposition, H.A. Schwarz (Rixena and Magoulès, 2007). Domain decomposition method is also the underlying paradigm of substructuring methods developed in the sixties, which aim at reducing the dimension of models in structural analysis by applying static condensation-type techniques to subdomains.

Other than static condensation, Farhat and Roux (1991a); Farhat (1991); Farhat and Geradin (1992) proposed FETI (Finite Element Tearing and Interconnecting) method for domain decomposition analysis. In FETI method, Lagrange multipliers are introduced to enforce compatibility at the interface nodes. Rigid body modes are eliminated in parallel from each local problem and a direct scheme is applied concurrently to all subdomains in order to recover each partial local solution. The contributions of these modes are then related to the Lagrange multipliers through an orthogonality condition. This FETI method has been shown that it can deliver high efficiency for parallel implicit transient simulations in structural mechanics (Crivelli and Farhat, 1993).

Domain decomposition itself has become a active topic as parallel processing techniques receive much more attention in mathematics and engineering world during recent years. Domain decomposition
was revived as a natural paradigm for parallel solvers (Rixena and Magoulès, 2007). Many papers have discussed two algorithms that are currently receiving much research effort, namely the FETI-DP (or Dual Primal Finite Element Tearing and Interconnecting) method and the even more recent BDDC (or Balancing Domain Decomposition by Constraints).

FETI-DP is the third generation FETI method (Bavestrello et al., 2007) developed for the fast, scalable, and domain-decomposition-based iterative solution of symmetric systems of equations arising from the finite element (FE) discretization of static, dynamic, structural and acoustic problems (Farhat et al., 2001, 2000).

BDDC, on the other hand, derives its formulation from substructuring method by enforcing constraints associated with disjoint sets of nodes on substructure boundaries using constrained energy minimization concepts (Dohrmann, 2003; Mandel and Dohrmann, 2003).

An early endeavor on dynamic computational load balancing was presented by McKenna (1997). Limited number of examples show that run time, dynamic computational load balancing can indeed improve parallel program performance in some cases, particularly when nonlinearities are involved.

Although many works have been presented on domain decomposition methods, the most popular methods such as FETI-type and BDDC all stem from the root of subdomain interface constraints handling. The merging of iterative solving with domain decomposition-type preconditioning is promising as shown by many researchers (Pavarino, 2007; Li and Widlund, 2007). Schwarz-type preconditioners for parallel domain decomposition system solving have also shared part of the spotlight (Hwang and Cai, 2007; Sarkis and Szyld, 2007).

In solid finite element methods, it has been assumed that the equation solving is the most computational expensive part so it is totally reasonable that all focus has been set on equation solver during the past decades.

Work presented in this chapter, however, has originated from the observation that for highly nonlinear materials, the constitutive level computation can be at least equally costly as equation solving, if not more expensive. The novelty of this chapter is to break out of the existing substructuring or FETI frameworks to further address the fundamental load balance issue of parallel computing. Namely, in order to achieve better parallel performance, we want to keep all processors equally busy. Load imbalance issue resulted from nonlinear constitutive level computations is too important to be neglected. This chapter proposes the Plastic Domain Decomposition algorithm which focuses on adaptive load balancing operation for nonlinear finite elements.

From the implementation point of view, for mesh-based scientific computations, domain decomposition corresponds to the problem of mapping a mesh onto a set of processors, which is well defined as a graph partitioning problem (Schloegel et al., 1999).
Formally, the graph partitioning problem is as follows. Given a weighted, undirected graph $G = (V; E)$ for which each vertex and edge has an associated weight, the $k$-way graph partitioning problem is to split the vertices of $V$ into $k$ disjoint subsets (or subdomains) such that each subdomain has roughly an equal amount of vertex weight (referred to as the balance constraint), while minimizing the sum of the weights of the edges whose incident vertices belong to different subdomains (i.e., the edge-cut).

In computational solid mechanics, the element graph is naturally used in parallel finite element method due to the fact that elemental operation forms the basis of finite element method. On the other hand, for material nonlinearity simulations, the element calculations represent the most computationally expensive part. In order to facilitate consistent interfaces for computational load measuring and data migration, element graph has been utilized as fundamental graph structure in this chapter, although it has been shown that the node-graph can be used as well for structure dynamics problem and the element-cut partitioning can make certain algorithms simpler (Krysl and Bittnar, 2001).

The graph partitioning problem is known to be NP-complete. Therefore, generally it is not possible to compute optimal partitioning for graphs of interesting size in a reasonable amount of time. Various heuristic approaches have been developed, which can be classified as either geometric, combinatorial, spectral, combinatorial optimization techniques, or multilevel methods (Dongarra et al., 2003).

In finite element simulations involving nonlinear material response, static graph partitioning mentioned above does not guarantee even load distribution among processors. Plastification introduces work load that is much heavier than pure elastic computation. So for this kind of multiphase simulation, adaptive computational load balancing scheme has to be considered to keep all processing units equally busy as much as possible. Traditional static graph partitioning algorithm is not adequate to do multiphase partition/repartitioning. A parallel multilevel graph partitioner has been introduced in this research to achieve dynamic load balancing for inelastic finite element simulations.

In this chapter, the algorithm of Plastic Domain Decomposition (PDD) is proposed. The adaptive multi-level graph partitioning kernel of the PDD algorithm is implemented through the ParMETIS interface. Studies are performed to extract optimal algorithmic parameters for our specific applications.

### 110.3.2 Inelastic Parallel Finite Element

The distinct feature of inelastic (elastic-plastic) finite element computations is the presence of two iteration levels. In a standard displacement based finite element implementation, constitutive driver

---

1 The complexity class NP is the set of decision problems that can be solved by a non-deterministic Turing machine in polynomial time. The NP-complete problems are the most difficult problems in NP ("non-deterministic polynomial time") in the sense that they are the smallest subclass of NP that could conceivably remain outside of P, the class of deterministic polynomial-time problems, [http://en.wikipedia.org/wiki/NP-complete](http://en.wikipedia.org/wiki/NP-complete)
at each Gauss point iterates in stress and internal variable space, computes the updated stress state, constitutive stiffness tensor and delivers them to the finite element functions. Finite element functions then use the updated stresses and stiffness tensors to integrate new (internal) nodal forces and element stiffness matrix. Then, on global level, nonlinear equations are iterated on until equilibrium between internal and external forces is satisfied within some tolerance.

- **Elastic Computations**
  In the case of elastic computations constitutive driver has a simple task of computing increment in stresses ($\Delta \sigma_{ij}$) for a given deformation increment ($\Delta \epsilon_{kl}$), through a closed form equation ($\Delta \sigma_{ij} = E_{ijkl} \Delta \epsilon_{kl}$). It is important to note that in this case the amount of work per Gauss point is known in advance. The amount of computational work is the same for every integration point. If we assume the same number of integration points per element, it follows that the amount of computational work is the same for each element and it is known in advance.

- **Elastic-Plastic Computations**
  On the other hand, for elastic-plastic problems, for a given incremental deformation the constitutive driver is iterating in stress and internal variable space until consistency condition is satisfied ($F = 0$). The number of iterations is not known in advance. Initially, all Gauss points are in elastic range, but as we incrementally apply loads, the plastic zones develop. For Gauss points still in elastic range, there are no iterations, the constitutive driver just computes incremental stresses from closed form solution. Computational load will increase significantly for integration of constitutive equations in plastic range. In particular, constitutive level integration algorithms for soils, concrete, rocks, foams and other granular materials are very computationally demanding. More than 70% of wall clock time during an elastic-plastic finite element analysis is spent in constitutive level iterations. This is in sharp contrast with elastic computations where the dominant part is solving the system of equations which consumes about 80% of run time. The extent of additional, constitutive level iterations is not known before the actual computations are over. In other words, the extent of elastic-plastic domain is not known ahead of time.

The traditional preprocessing type of Domain Decomposition method (also known as topological DD) splits domain based on the initial geometry and assigns roughly the same number of elements to every computational node and minimizes the size of subdomain boundaries. This approach might result in serious computational load imbalance for elastic-plastic problems. For example one domain might be assigned all of the elastic-plastic elements and spend large amount of time in constitutive level iterations. The other domains will have elements in elastic state and thus spend far less computational time in computing stress increments. This results in program having to wait
for the slowest domain (the one with large number of elastic-plastic finite elements) to complete
costitutive level iterations and only proceed with global system iterations after that.

This illustrates a two-fold challenge with computational load balancing for inelastic simulations in
mechanics. These two challenges is described below in some more detail.

110.3.2.1 Adaptive Computation

First, these computations are dynamic in nature. That is, the structure of elastic and elastic-plastic
domains changes dynamically and unpredictably during the course of the computation. For this reason,
a static decomposition computed as a pre-processing step is not sufficient to ensure the computational
load-balance of the entire computation. Instead, periodic computational load-balancing is required
during the course of the computation. The problem of computing a dynamic decomposition shares
the same requirements as that of computing the initial decomposition (i.e., balance the mesh elements
and minimize the inter-processor communications), while also requiring that the cost associated with
redistributing the data in order to balance the computational load is minimized. This last requirement
prevents us from simply computing a whole new static partitioning from scratch each time computational
load-balancing is required.

Often, the objective of minimizing the data redistribution cost is at odds with the objective of min-
imizing the inter-processor communications. For applications in which the computational requirements
of different regions of the domain change rapidly, or the amount of state associated with each element is
relatively high, minimizing the data redistribution cost is preferred over minimizing the communications
incurred during parallel processing.

For applications in which computational load-balancing occurs very infrequently, the key objective of
a load-balancing algorithm is in obtaining the minimal inter-processor communications. For many appli-
cation domains, it is straightforward to select a primary objective to minimize (i.e., minimize whichever
cost dominates). However, one of the key issues concerning the elastic-plastic computation is that the
number of iterations between computational load-balancing phases is both unpredictable and dynamic.
For example, in the case of static problems, zones in the 3D solid may become plastic and then unload
to elastic (during increments of loading) so that the extent of plastic zone is changing. The change
can be both slow and rapid. Slow change usually occurs during initial loading phases, while the later
deformation tends to localize in narrow zones rapidly and the rest of the solid unloads rapidly (becomes
elastic again). The narrow, localized zone has heavy computational load on the constitutive level (in each
integration point within elements). Similar phenomena is observed in seismic soil-structure interaction
computations where stiff structure interacts with soft soil and elastic and elasto-plastic zones change
significantly during loading cycles. In this type of computation, it is extremely difficult to select the type of computational load-balancing algorithm to employ. Furthermore, the preferred computational load-balancing algorithm is liable to change during the course of the computation, and so the selection must be made dynamically.

### 110.3.2.2 Multiphase Computation

The second challenge associated with computational load-balancing elastic-plastic computations in geomechanics is that these are two-phase computations. That is, elastic-plastic computations follow up the elastic computations. There is a synchronization phase between the computations, as only after the elastic computation is finished is it possible to check if the elastic-plastic computation is required for a given integration (Gauss) point within an element. For regions of the mesh in which this check indicates that the elastic-plastic computation is necessary, lengthy elastic-plastic computations are then performed. The existence of the synchronization step between the two phases of the computation requires that each phase be individually load balanced. That is, it is not sufficient to simply sum up the relative times required for each phase and to compute a decomposition based on this sum. Doing so may lead to some processors having too much work during the elastic computation (and so, these may still be working after other processors are idle), and not enough work during the elastic-plastic computation, (and so these may be idle while other processors are still working), and vice versa. Instead, it is critical that every processor have an equal amount of work from both of the phases of the computation.

### 110.3.2.3 Multiconstraint Graph Partitioning

Elastic-plastic FE computation can be understood as a two-phase calculation, which is also dynamic in nature. Traditional graph partitioning formulations are not adequate to ensure its efficient execution on high performance parallel computers. In this chapter very recent progresses from the graph partitioning algorithm research will be investigated. We need new adaptive graph partitioning formulations, which can compute adaptive partitioning-repartitionings that can satisfy an arbitrary number of balance constraints.

- **Static Graph Partitioning**
  Given a weighted, undirected graph $G = (V, E)$, for which each vertex and edge has an associated weight, the $k$-way graph partitioning problem is to split the vertices of $V$ into $k$ disjoint subsets (or subdomains) such that each subdomain has roughly an equal amount of vertex weight (referred to as the balance constraint), while minimizing the sum of the weights of the edges whose incident vertices belong to different subdomains (i.e., the edge-cut).
1. Geometric Techniques

Compute partitioning based solely on the coordinate information of the mesh nodes, without considering edge-cut. Popular methods include, Coordinate Nested Dissection (CND or Recursive Coordinate Bisection), Recursive Inertial Bisection (RIB), Space-Filling Curve techniques and Sphere-Cutting approach.

2. Combinatorial Techniques

Attempt to group together highly connected vertices whether or not these are near each other in space. That is combinatorial partitioning schemes compute a partitioning based only on the adjacency information of the graph; they do not consider the coordinates of the vertices. They tend to have lower edge-cuts but generally slower. Popular methods include, Levelized Nested Dissection (LND) and Kernighan-Lin/Fiduccia-Mattheyses (KL/FM) partitioning refinement algorithm, which needs an initial partition input to do swapping refinement.

3. Multilevel Schemes

The multilevel paradigm consists of three phases: graph coarsening, initial partitioning, and multilevel refinement. Firstly, we form coarse graph by collapsing together selected vertices of the input graph. After rounds of coarsening, we get coarsest graph, on which an initial bisection will be performed. Then the KL/FM algorithm can be used to refine the partition back to the finest graph.

The multilevel paradigm works well for two reasons. First, a good coarsening scheme can hide a large number of edges on the coarsest graph, which makes the task of computing high-quality partitioning easier. Second reason, incremental refinement schemes such as KL/FM become much more powerful in the multilevel context.

Popular algorithms include Multilevel Recursive Bisection and Multilevel k-Way Partitioning.

- Adaptive Graph Partitioning

For large scale elasto-plastic FE simulations, it is necessary to dynamically load-balance the computations as the analysis progresses due to unpredictable plastification inside the domain. This dynamic load balancing can be achieved by using a graph partitioning algorithm.

Adaptive graph partitioning shares most of the requirements and characteristics of static graph partitioning but also adds an additional objective. That is, the amount of data that needs to be redistributed among the processors in order to balance the load should be minimized. If the vertex weight represents the computational cost of the work carried by the vertex, another metric, size of the vertex needs to be considered as well, which reflects distribution cost of the vertex. Thus, the repartitioner should attempt to balance the partitioning with respect to vertex weight while...
minimizing vertex migration with respect to vertex size.

Different approaches are available. One can simply compute a new graph from scratch, so called
Scratch-Remap Repartitioner, which expectedly introduces more data redistribution than necessary.
Diffusion-Based Repartitioner attempt to minimize the difference between the original partitioning
and the final repartitioning by making incremental changes in the partitioning to restore balance.
This method has been an very active topic during recent years, Dongarra et al. (2003) gives
up-to-date review.

• Multiconstraint Graph Partitioning

We can see traditional graph partitioning typically balances only a single constraint (i.e., the vertex
weight) and minimizes only a single objective (i.e., the edge-cut). If we replace the vertex weight,
which is a single number, with a weight vector of size \( m \), then the problem becomes that of finding
a partitioning that minimizes the edge-cuts subject to the constraints that each of the \( m \) weights
is balanced across subdomains.

Multilevel graph partitioning algorithms for solving multiconstraint/multiobjective problems have
been very successful Schloegel et al. (1999). The software libraries METIS and ParMETIS are
widely used in computational mechanics research.

### 110.3.2.4 Adaptive PDD Algorithm

In this chapter, the Plastic Domain Decomposition (PDD) has been developed using multi-level, multi-
objective graph partitioning algorithm. This algorithm automatically monitors load balancing condition
and updates element graph structure accordingly as the simulation progresses. Element redistribution will
be triggered to achieve load balance when nonlinearity of materials brings down the parallel performance.

### 110.3.3 Adaptive Multilevel Graph Partitioning Algorithm

Kaypis and Kumar (1998) present a \( k \)-way multilevel partitioning algorithm whose run time is linear in
the number of edges \(|E|\) (i.e., \( O(|E|) \)); whereas the run time of multilevel recursive bisection schemes
is \( O(|E|\log k) \) for \( k \)-way partitioning. Kaypis and Kumar (1998) show that the proposed multilevel
partitioning scheme produces partitioning that are of comparable or better quality than those produced
by multilevel recursive bisection, while requiring substantially less time. This paradigm consists of three
phases: graph coarsening, initial partitioning, and multilevel refinement. In the graph coarsening phase, a
series of graphs is constructed by collapsing together selected vertices of the input graph in order to form
a related coarser graph. This newly constructed graph then acts as the input graph for another round of
graph coarsening, and so on, until a sufficiently small graph is obtained. Computation of the initial
bisection is performed on the coarsest (and hence smallest) of these graphs, and so is very fast. Finally, partition refinement is performed on each level graph, from the coarsest to the nest (i.e., original graph) using a KL/FM-type algorithm Dongarra et al. (2003). Figure 110.1 illustrates the multilevel paradigm. This algorithm is available in METIS Karypis and Kumar (1998d) which is used in this research to provide initial static partitioning.

![Multilevel K-way Partitioning](image)

Figure 110.1: Multilevel Graph Partitioning Scheme Karypis et al. (2003)

Adaptive graph repartitioning algorithm can be used to achieve dynamic load balancing of multiphase elastic-plastic finite element simulations. Adaptive graph partitioning differs from static graph partitioning algorithm in the sense that one additional objective has to be targeted. That is, the amount of data needs to be redistributed among the processors in order to balance the load should be minimized. In order to measure this redistribution cost, not only does the weight of a vertex, but also its size have to be considered. In our implementation for the purpose of this research, the vertex weight represents the computational load of each finite element, while the size reflects its redistribution cost. Thus, the application of adaptive graph partitioning algorithm aims at balancing the partitioning with respect to vertex weight while minimizing vertex migration with respect to vertex size.

A repartitioning of a graph can be obtained simply by partitioning a new graph from a scratch, which tends to bring much more unnecessary communications because the old distribution has not been taken
into account. Diffusion-based Repartitioner is more popular in which one attempts to minimize the difference between the original partitioning and the final repartitioning by making incremental changes in the partitioning to restore balance. Dongarra et al. (2003) gives a comprehensive review on this subject. Adaptive repartitioning is available in ParMETIS Karypis et al. (2003) and Jostle Warshaw (1998). The former is chosen in this research considering the fact that ParMETIS provides seamless interface for METIS 4.0 which makes the comparison between static and adaptive partitioning schemes more consistent.

PARMETIS is an MPI-based parallel library that implements a variety of algorithms for partitioning and repartitioning unstructured graphs and for computing fill-reducing orderings of sparse matrices Karypis et al. (2003). PARMETIS is particularly suited for parallel numerical simulations involving large unstructured meshes. In this type of computation, PARMETIS dramatically reduces the time spent in communication by computing mesh decompositions such that the numbers of interface elements are minimized. The algorithms in PARMETIS are based on the multilevel partitioning and fill-reducing ordering algorithms that are implemented in the widely-used serial package METIS Karypis and Kumar (1998c). However, PARMETIS extends the functionality provided by METIS and includes routines that are especially suited for parallel computations and large-scale numerical simulations. In particular, PARMETIS provides the following functionality Karypis et al. (2003):

- Partition unstructured graphs and meshes.
- Repartition graphs that correspond to adaptively refined meshes.
- Partition graphs for multi-phase and multi-physics simulations.
- Improve the quality of existing partitioning.
- Compute fill-reducing orderings for sparse direct factorization.
- Construct the dual graphs of meshes.

Both METIS and PARMETIS are used in this research. METIS routines are called to construct static partitioning for commonly used one-step static domain decomposition, while adaptive load-balancing is achieved by calling PARMETIS routines regularly during the progress of nonlinear finite element simulations.

Adaptive load-balancing through domain repartitioning is a multi-objective optimization problem, in which repartitionings should minimize both the inter-processor communications incurred in the iterative mesh-based computation and the data redistribution costs required to balance the load. PARMETIS
provides the routine `ParMETIS_V3_AdaptiveRepart` for repartitioning the previous unbalanced computational domain. This routine assumes that the existing decomposition is well distributed among the processors, but that (due to plastification of certain nonlinear elements) this distribution is poorly load balanced.

Figure 110.2: A diagram illustrating the execution of adaptive scientific simulations on high performance parallel computers Schloegel et al. (1999)

Figure 110.2 Schloegel et al. (2000) shows common steps involved in the execution of adaptive mesh-based simulations on parallel computers. Initially, the mesh is equally distributed on different processors. As all elements are elastic at the very beginning (carrying the same amount of elemental calculation work), computation load balance can be guaranteed with a even distribution. A number of iterations of the simulation are performed in parallel, after which plasticity occurs in certain nonlinear elements thus introducing some amount of load imbalance. A new partitioning based on the unbalanced domain is computed to re-balance the load, and then the mesh is redistributed among the processors, respectively. The simulation can then continue for another number of iterations until either more mesh adaptation is required or the simulation terminates.

If we consider each round of executing a number of iterations of the simulation, mesh adaptation, and load-balancing to be an epoch, then the run time of an epoch can be described by, Schloegel et al.
(2000)

\[(t_{\text{comp}} + f(|E_{\text{cut}}|))n + t_{\text{repart}} + g(|V_{\text{move}}|)\]  

(110.3)

where \(n\) is the number of iterations executed, \(t_{\text{comp}}\) is the time to perform the computation for a single iteration of the simulation, \(f(|E_{\text{cut}}|)\) is the time to perform the communications required for a single iteration of the simulation, and \(t_{\text{repart}}\) and \(g(|V_{\text{move}}|)\) represent the times required to compute the new partitioning and to redistribute the data. Here, the inter-processor communication time is described as a function of the edge-cut of the partitioning and the data redistribution time is described as a function of the total amount of data that is required to be moved in order to realize the new partitioning. Adaptive repartitioning affects all of terms in Equation 110.3. How well the new partitioning is balanced influences \(t_{\text{comp}}\). The inter-processor communications time is dependent on the edge-cut of the new partitioning. The data redistribution time is dependent on the total amount of data that is required to be moved in order to realize the new partitioning. It is critical for adaptive partitioning schemes to minimize both the edge-cut and the data redistribution when computing the new partitioning. Viewed in this way, adaptive graph partitioning is a multi-objective optimization problem.

There are various approaches how to handle this dual-objective problem. In general, two approaches have primarily been taken when designing adaptive partitioners. Schloegel et al. (2000) gives a comprehensive review on this topic. The first approach is to attempt to focus on minimizing the edge-cut and to minimize the data redistribution only as a secondary objective. This family of methods can be called _scratch-remap_ repartitioner. These use some type of state-of-the-art graph partitioner to compute a new partitioning from scratch and then attempt to intelligently remap the subdomain labels to those of the original partitioning in order to minimize the data redistribution costs. Since a state-of-the-art graph partitioner is used to compute the partitioning, the resulting edge-cut tends to be extremely good. However, since there is no guarantee as to how similar the new partitioning will be to the original partitioning, data redistribution costs can be high, even after remapping. The second approach is to focus on minimizing the data redistribution cost and to minimize the edge-cut as a secondary objective, or so-called _diffusion-based_ repartitioner. These schemes attempt to perturb the original partitioning just enough so as to balance it. This strategy usually leads to low data redistribution costs, especially when the partitioning is only slightly imbalanced. However, it can result in higher edge-cuts than scratch-remap methods because perturbing a partitioning in order to balance it also tends to adversely affect its quality.

These two types of repartitioner allow the user to compute partitioning that focus on minimizing either the edge-cut or the data redistribution costs, but give the user only a limited ability to control the tradeoffs among these objectives. This control of the tradeoffs is sufficient if the number of iterations
that a simulation performs between load-balancing phases (i.e. the value of \( n \) in Equation 110.3 ) is either very high or very low. However, when \( n \) is neither very high nor very low, neither type of scheme precisely minimizes the combined costs of \( f(E_{\text{cut}}|n) \) and \( g(|V_{\text{move}}|) \). Another disadvantage exists for applications in which \( n \) is difficult to predict or those in which \( n \) can change dynamically throughout the course of the computation. As an example, one of the key issues concerning the elastic-plastic soil-structure interaction computations required for earthquake simulation is that the number of iterations between load-balancing phases is both unpredictable and dynamic. Here, zones in the 3D solid may become plastic and then unload (during increments of loading) so that the extent of the plastic zone is changing. The change can be both slow and rapid. Slow change usually occurs during initial loading phases, while the later deformation tends to localize in narrow zones rapidly and the rest of the solid unloads rapidly (becomes elastic again) Jeremić and Xenophontos (1999).

Schloegel et al. (2000) presents a parallel adaptive repartitioning scheme (called the Unified Repartitioning Algorithm) for the dynamic load-balancing of scientific simulations that attempts to solve the precise multi-objective optimization problem. By directly minimizing the combined costs of \( f(E_{\text{cut}}|n) \) and \( g(|V_{\text{move}}|) \), the proposed scheme is able to gracefully tradeoff one objective for the other as required by the specific application. The paper shows that when inter-processor communication costs are much greater in scale than data redistribution costs, the proposed scheme obtains results that are similar to those obtained by an optimized scratch-remap repartitioner and better than those obtained by an optimized diffusion-based repartitioner. When these two costs are of similar scale, the scheme obtains results that are similar to the diffusive repartitioner and better than the scratch-remap repartitioner. When the cost to perform data redistribution is much greater than the cost to perform inter-processor communication, the scheme obtains better results than the diffusive scheme and much better results than the scratch-remap scheme. They also show in the paper that the Unified Repartitioning Algorithm is fast and scalable to very large problems.

### 110.3.3.1 Unified Repartitioning Algorithm

A key parameter used in Unified Repartitioning Algorithm (URA) is the Relative Cost Factor (RCF). This parameter describes the relative times required for performing the inter-processor communications incurred during parallel processing and to perform the data redistribution associated with balancing the load. Using this parameter, it is possible to unify the two minimization objectives of the adaptive graph partitioning problem into the unified cost function

\[
|E_{\text{cut}}| + \alpha|V_{\text{move}}|
\]  

(110.4)
where $\alpha$ is the Relative Cost Factor, $|E_{\text{cut}}|$ is the edge-cut of the partitioning, and $|V_{\text{move}}|$ is the total amount of data redistribution. The Unified Repartitioning Algorithm attempts to compute a repartitioning while directly minimizing this cost function.

The Unified Repartitioning Algorithm is based upon the multilevel paradigm that is illustrated in Figure 110.1, which can be described as three phases: graph coarsening, initial partitioning, and uncoarsening/refinement Schloegel et al. (2000). In the graph coarsening phase, coarsening is performed using a purely local variant of heavy-edge matching. That is, vertices may be matched together only if they are in the same subdomain on the original partitioning. This matching scheme has been shown to be very effective at helping to minimize both the edge-cut and data redistribution costs and is also inherently more scalable than global matching schemes.

### 110.3.3.2 Study of ITR in ParMETIS

The RCF in the URA implementation controls the tradeoff between two objectives, minimizing data redistribution cost or edge-cut. In our application, ParMETIS library has been linked to a MOSS (Modified OpenSees Services) analysis model to facilitate the partitioning/adaptive repartitioning scheme. The RCF is defined as a single parameter ITR in ParMETIS Karypis et al. (2003). This parameter describes the ratio between the time required for performing the inter-processor communications incurred during parallel processing compared to the time to perform the data redistribution associated with balancing the load. As such, it allows us to compute a single metric that describes the quality of the repartitioning, even though adaptive repartitioning is a multi-objective optimization problem. As recommended by Karypis et al. (2003), appropriate values to pass for the ITR Factor parameter can be determined depending on the times required to perform

1. all inter-processor communications that have occurred since the last repartitioning, and

2. the data redistribution associated with the last repartitioning/load balancing phase.

Simply divide the first time measurement by the second time measurement. The result is the correct ITR Factor. In case these times cannot be ascertained (e.g., for the first repartitioning/load balancing phase), Karypis et al. (2003) suggests that values between 100 and 1000 work well for a variety of situations. By default ITR is between 0.001 and 1000000. If ITR is set high, a repartitioning with a low edge-cut will be computed. If it is set low, a repartitioning that requires little data redistribution will be computed.
110.4 Performance Studies on PDD Algorithm

110.4.1 Introduction

In this chapter, parallel performance of the proposed PDD algorithm is thoroughly investigated. There are two major focuses for the timing analysis. Firstly we want to see how much performance gain we can have by introducing the PDD algorithm into inelastic finite element calculations. Secondly, we also want to show how scalable the proposed PDD algorithm is.

As our final objective is to apply PDD in large scale SFSI finite element simulations, finite element models of SFSI have been set up to study the parallel performance of the PDD based parallel program. Implicit constitutive integration scheme Jeremić and Sture (1997) has been used to expose the load imbalance by plasticity calculation. Only continuum element has been studied due to the fact they can be easily visualized to obtain partition and/or repartition figures.

Distributed memory Linux/Unix clusters are major platforms used in this chapter for speed up analysis.

110.4.2 Parallel Computers

Performance measurement has been carried out on two SMP-based clusters.

- **IBM eServer p655**
  
  The DataStar IBM eServer p655 cluster consists of 176 8-way P655+ nodes at San Diego Supercomputer Center. System configuration is shown in Fig 110.3. The network benchmark is shown in Table 110.1.

<table>
<thead>
<tr>
<th></th>
<th>MPI Latencies (μsec)</th>
<th>Bandwidth (MBs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intra-node</strong></td>
<td>3.9</td>
<td>3120.4</td>
</tr>
<tr>
<td><strong>Inter-node</strong></td>
<td>7.65</td>
<td>1379.1</td>
</tr>
</tbody>
</table>

- **TeraGrid IA-64 Intel-Based Linux Cluster**

  The TeraGrid project was launched by the the National Science Foundation with $53 million in funding to four sites: the National Center for Supercomputing Applications (NCSA) at the University of Illinois, Urbana-Champaign, the San Diego Supercomputer Center (SDSC) at the University of California, San Diego, Argonne National Laboratory in Argonne, IL, and Center for Advanced Computing Research (CACR) at the California Institute of Technology in Pasadena.
SDSC’s TeraGrid cluster currently consists of 256 IBM cluster nodes, each with dual 1.5 GHz Intel Itanium 2 processors, for a peak performance of 3.1 teraflops. The nodes are equipped with four gigabytes (GBs) of physical memory per node. The cluster is running SuSE Linux and is using Myricom’s Myrinet cluster interconnect network. Table 110.2 shows the technical configuration of the IA64 cluster, on which the second part of the performance study has been done.

### 110.4.3 Soil-Foundation Interaction Model

A soil-shallow-foundation interaction model as shown in Figure 110.4 has been set up to study the parallel performance. 3D brick element with 8 integration (Gaussian) points is used. The soil is modeled by Template3D elasto-plastic material model (Drucker-Prager model with Armstrong Frederick nonlinear kinematic hardening rule) and linear elasticity is assumed for the foundation. More advanced constitutive laws can be applied through Template3D model although the model used here suffices the purpose of this research to show repartitioning triggered by plastification. It is shown in this research that the speedup by adaptive load balancing is significant even for seemingly simple constitutive model. The material properties are shown in Table 110.3 and the vertical loading is applied at 5.0kN increments. The performance analysis has been carried out on DataStar supercomputer at San Diego Supercomputing Center (P655+ 8-way nodes).
Table 110.2: Technical Information of IA64 TeraGrid Cluster at SDSC

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Linux Cluster</td>
</tr>
<tr>
<td>Access Nodes</td>
<td>* quad-processor</td>
</tr>
<tr>
<td></td>
<td>* ECC SDRAM memory: 8 GB</td>
</tr>
<tr>
<td></td>
<td>* 2 nodes (8 processors)</td>
</tr>
<tr>
<td>Compute Nodes</td>
<td>* dual-processor</td>
</tr>
<tr>
<td></td>
<td>* ECC SDRAM memory: 4 GB</td>
</tr>
<tr>
<td></td>
<td>* 262 nodes (524 processors)</td>
</tr>
<tr>
<td>Processor</td>
<td>* Intel Itanium 2, 1.5 GHz</td>
</tr>
<tr>
<td></td>
<td>* Integrated 6 MB L3 cache</td>
</tr>
<tr>
<td></td>
<td>* Peak performance 3.1 Tflops</td>
</tr>
<tr>
<td>Network Interconnect</td>
<td>Myrinet 2000, Gigabit Ethernet, Fiber Channel</td>
</tr>
<tr>
<td>Disk</td>
<td>1.7 TB of NFS, 50 TB of GPFS (Parallel File System)</td>
</tr>
<tr>
<td>Operating System</td>
<td>Linux 2.4-SMP (SuSE SLES 8.0)</td>
</tr>
<tr>
<td>Compilers</td>
<td>* Intel: Fortran77/90/95 C C++</td>
</tr>
<tr>
<td></td>
<td>* GNU: Fortran77 C C++</td>
</tr>
<tr>
<td>Batch System</td>
<td>Portable Batch System (PBS) with Catalina Scheduler</td>
</tr>
</tbody>
</table>

Table 110.3: Material Constants for Soil-Foundation Interaction Model

<table>
<thead>
<tr>
<th>Soil</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic modulus</td>
<td>$E = 17400$ kPa</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>$\nu = 0.35$</td>
</tr>
<tr>
<td>Friction angle</td>
<td>$\phi = 37.1^\circ$</td>
</tr>
<tr>
<td>Cohesion</td>
<td>$c = 0$</td>
</tr>
<tr>
<td>Isotropic Hardening</td>
<td>Linear</td>
</tr>
<tr>
<td>Kinematic Hardening</td>
<td>A/F nonlinear ($h_a = 116.0$, $C_r = 80.0$)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Foundation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic modulus</td>
<td>$E = 21$ GPa</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>$\nu = 0.2$</td>
</tr>
</tbody>
</table>
110.4.4 Numerical Study for ITR

As described in Section 110.3.3.2, the parameter ITR in ParMETIS describes the ratio between the time required for performing the inter-processor communications incurred during parallel processing compared to the time to perform the data redistribution associated with balancing the load. It acts like a switch on algorithmic approaches of ParMETIS repartitioning kernel. With ITR factor being very small, the ParMETIS tends to do that repartitioning which can minimize data redistribution cost. If the ITR factor is set to be very large, ParMETIS tends to minimize edge-cut of the final repartition.

In parallel design of PDD, if repartitioning is necessary to achieve load balance after each load increment, the whole AnalysisModel McKenna (1997) has to be wiped off thus a new analysis container
can be defined to reload subsequent analysis steps. The data redistribution cost can be much higher than communication overhead only. In order to determine an adequate ITR value for our application, preliminary study needs to be performed to investigate the effectiveness of the URA. In this research, two extreme values of the ITR (0.001 and 1,000,000) are prescribed and then parallel analysis is carried out on 2, 4 and 8 processors to see how the partition/repartition algorithm behaves. Two soil-structure interaction models as shown in Figure 110.6 have been used in this parametric study. Timing data and partition figures have been collected to investigate the performance of different approaches. The one that tends to bring better performance will be adopted in subsequent parallel analysis for prototype 3D soil structure interaction problems. Figure 110.7 to Figure 110.12 shows the initial partition and final repartition figures for two different types of algorithms. With ITR factor to be very small, the URA tends to present results that minimize data redistribution cost, in which diffusive repartitioning approach is used. On the other hand, if the ITR factor is set to be very large, then the URA algorithm tends to give repartitioning with lowest edge cut but with considerably higher data redistribution cost.
Figure 110.5: FE Models (1,968 Elements, 7,500 DOFs) for Studying Soil-Foundation Interaction Problems

Figure 110.6: FE Models (4,938 Elements, 17,604 DOFs) for Studying Soil-Foundation Interaction Problems
Figure 110.7: Partition and Repartition on 2 CPUs ($\text{ITR}=1\times10^{-3}$, Imbal. Tol. 5%), FE Model (1,968 Elements, 7,500 DOFs)

Figure 110.8: Partition and Repartition on 2 CPUs ($\text{ITR}=1\times10^{6}$, Imbal. Tol. 5%), FE Model (1,968 Elements, 7,500 DOFs)
Figure 110.9: Partition and Repartition on 4 CPUs (ITR=1e-3, Imbal. Tol. 5%), FE Model (1,968 Elements, 7,500 DOFs)

Figure 110.10: Partition and Repartition on 4 CPUs (ITR=1e6, Imbal. Tol. 5%), FE Model (1,968 Elements, 7,500 DOFs)
Figure 110.11: Partition and Repartition on 7 CPUs (ITR=1e-3, Imbal. Tol. 5%), FE Model (1,968 Elements, 7,500 DOFs)

Figure 110.12: Partition and Repartition on 7 CPUs (ITR=1e6, Imbal. Tol. 5%), FE Model (1,968 Elements, 7,500 DOFs)
Figure 110.13: Partition and Repartition on 7 CPUs (ITR=1e-3, Imbal. Tol. 5%), FE Model (4,938 Elements, 17,604 DOFs)

Figure 110.14: Partition and Repartition on 7 CPUs (ITR=1e6, Imbal. Tol. 5%), FE Model (4,938 Elements, 17,604 DOFs)
Figure 110.15: Timing Data of ITR Parametric Studies (1,968 Elements, 7,500 DOFs, Imbal. Tol. 5%)
Figure 110.16: Relative Speedup of ITR=1e-3 over ITR=1e6 (1,968 Elements, 7,500 DOFs, Imbal. Tol. 5%)
Figure 110.17: Timing Data of ITR Parametric Studies (4,938 Elements, 17,604 DOFs, Imbal. Tol. 5%)
Figures 110.15, 110.16 and 110.17 show the speedup data of parametric study on ITR factors. The purpose is to expose the the more efficient approach to do repartitioning for our specific parallel SFSI simulations, either scratch-remap approach ($ITR = 1e6$) or diffusive approach ($ITR = 1e-3$). Through the study of this chapter, some conclusions can be drawn.

1. Smaller value of ITR (1e-3) outperforms larger value (1e6). The performance gain is up to 22.1% for 7 processors. As the model gets larger, the speedup tends to get better.

2. With small ITR value, the URA algorithm tends to give results for diffusive partition/repartitioning scheme, which is good for performance for our application in overall due to the fact that the overhead associated with data redistribution in this research is very high. Diffusive approach minimizes possible data movement thus delivers better performance. The drawback is the diffusive approach typically gives very bad or even disconnected graphs with very high edge-cut as shown in Figures 110.7, 110.9 and 110.11. So careful attention must be paid to these graph structures when programing the finite element calculation. In this sense, the diffusive algorithm is not as robust as scratch/remapping. One very important observation was that repetitive repartitionings tend to yield totally ill-connected graph.

3. With large ITR value, the URA algorithm adopts the scratch/remapping scheme which inevitably introduce huge data redistribution cost. But this approach gives high quality graph and the integrity of original graph is well preserved as shown Figures 110.8, 110.10 and 110.12. This will be of great meaning for parallel finite element method based on substructure-type methods. Another important observation was, the scratch/remapping approach performed much more repartitionings than diffusive approach for same analysis. Repetitive repartitionings by scratch/remapping method tends to totally migrate all elements out of their initial partitioning and repartitioning never stops even though the computation is stabilized (in the sense of formation of plastic zones). This also explains in part why the diffusive approach can substantially outperform scratch/remapping.

4. Based on the timing analysis performed in this chapter, ITR=1e-3 is the best choice that brings substantially better performance over large ITR values. With the increase of number of processing units or the model size, the performance gain is more significant as shown in Figures 110.15, 110.16 and 110.17. Robustness of the diffusive approach has not caused much trouble in our application.

110.4.5 Parallel Performance Analysis

Timing routines have been implemented in PDD (MOSS and other used libraries, such as Template3DEP/NewTemplate3DEP) to study the parallel performance. The preprocessing unit, like reading model data from file, has not
been timed so the speed up here reflects only algorithmic gain by graph partitioning. In the current phase of this research, the equation solving problem has not been addressed yet. More meaningful perspective would be to consider performance gains by simply switching from plain graph partitioning to adaptive graph partitioning, which is also the basic aim of this research. As we can see from the results below, adaptive graph partitioning improves the overall performance of elasto-plastic finite element computations. The partitioning/repartitioning overhead has been minimized by using parallel partitioner.

As stated in previous sections of this chapter, there are a couple of key parameters that control performance of the adaptive load balancing algorithm. One is the ITR factor, and the other is the computational load imbalance tolerance.

1. **ITR** is the key parameter which determines the algorithmic approach of the adaptive load balancing scheme. Depending on different applications and network interconnections, this value can be set to very small (0.001) or very large (up to 1,000,000) and algorithm focus will be set to minimizing data redistribution or edge-cut respectively as explained in previous sections.

2. **Computational Imbalance Load Tolerance** is the other key factor affecting greatly the overall performance of the whole application codes. Basically speaking, with larger finite element model, the tolerance should be set higher due to the fact that data redistribution and subsequent analysis-restarting overhead can be substantially higher as the finite element model size increases.

The performance tunings on ITR factor tend to yield consistent results as stated previously that smaller ITR (0.001) brings better performance over large ITR values. Diffusive repartitioning algorithm outperforms scratch/remapping in our application.

While on the other hand, tuning on load imbalance studies has been more illusive. The first conclusion is that load imbalance tolerance larger than 5% was not able to work robustly as the size of finite element model increases in the application study of this chapter.

Detailed parametric studies have been performed on DataStar IBM Power4 and IA64 Intel clusters to indicate the effectiveness of the proposed adaptive PDD algorithm. Models with different sizes have been tested on various number of processors to show the scalability of computational performance. All results will be compared with static one-step Domain Decomposition approach to investigate the advantage of proposed PDD algorithm in nonlinear elastic-plastic finite element calculations.

In the following sections, timing data and partition/repartition figures will be presented and results will be discussed at the end of this chapter.

### 110.4.5.1 Soil-Foundation Model with 4,035 DOFs

The partition/repartition figures by PDD have been shown in Figure 110.18, 110.19, 110.20.
Figure 110.18: 4,035 DOFs Model, 2 CPUs, ITR=1e-3, Imbal Tol 5%, PDD Partition/Repartition

Figure 110.19: 4,035 DOFs Model, 4 CPUs, ITR=1e-3, Imbal Tol 5%, PDD Partition/Repartition
Figure 110.20: 4,035 DOFs Model, 8 CPUs, ITR=1e-3, Imbal Tol 5%, PDD Partition/Repartition

Figure 110.21: Timing Data of Parallel Runs on 4,035 DOFs Model, ITR=1e-3, Imbal Tol 5%
Table 110.4: Test Cases of Performance Studies

<table>
<thead>
<tr>
<th>Model Sizes (DOF)</th>
<th>4,035, 17,604, 32,091, 68,451</th>
</tr>
</thead>
<tbody>
<tr>
<td># of CPUs</td>
<td>3, 5, 7, 16, 32, 64</td>
</tr>
<tr>
<td>ITR Factors</td>
<td>0.001, 1,000,000</td>
</tr>
<tr>
<td>Imbalance Tolerance</td>
<td>5%, 10%, 20%</td>
</tr>
</tbody>
</table>

Figure 110.22: Absolute Speedup Data of Parallel Runs on 4,035 DOFs Model, ITR=1e-3, Imbal Tol 5%
Figure 110.23: Relative Speedup of PDD over Static DD on 4,035 DOFs Model, ITR=1e-3, Imbal Tol 5%
110.4.5.2 Soil-Foundation Model with 4,938 Elements, 17,604 DOFs

This is the same model as described before but with more elements as shown in 110.24. Timing data has been collected to indicate performance gains by adaptive load balancing. Partition and repartition figures are shown from Figure 110.28 to 110.30. The partition/repartition figures by PDD have been shown in Figure 110.28, 110.29, 110.30.
Figure 110.25: Timing Data of Parallel Runs on 4,938 Elements, 17,604 DOFs Model, ITR=1e-3, Imbal Tol 5%
Figure 110.26: Absolute Speedup Data of Parallel Runs on 4,938 Elements, 17,604 DOFs Model, ITR=1e-3, Imbal Tol 5%
Figure 110.27: Relative Speedup of PDD over Static DD on 4,938 Elements, 17,604 DOFs Model, ITR=1e-3, Imbal Tol 5%

Figure 110.28: 4,938 Elements, 17,604 DOFs Model, 2 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%
Figure 110.29: 4,938 Elements, 17,604 DOFs Model, 4 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%

Figure 110.30: 4,938 Elements, 17,604 DOFs Model, 8 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%
110.4.5.3 Soil-Foundation Model with 9,297 Elements, 32,091 DOFs

The mesh is shown in Figure 110.31. Speed up results are shown from Figure 110.32 to Figure 110.34. Partition and repartition figures are shown from Figure 110.35 to Figure 110.39.

Figure 110.31: Finite Element Model of Soil-Foundation Interaction (9,297 Elements, 32,091 DOFs)
Figure 110.32: Timing Data of Parallel Runs on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 5%
Figure 110.33: Absolute Speedup Data of Parallel Runs on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 5%
Figure 110.34: Relative Speedup of PDD over Static DD on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 5%

Figure 110.35: 9,297 Elements, 32,091 DOFs Model, 3 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%
Figure 110.36: 9,297 Elements, 32,091 DOFs Model, 5 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%

Figure 110.37: 9,297 Elements, 32,091 DOFs Model, 7 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%
Figure 110.38: 9,297 Elements, 32,091 DOFs Model, 16 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%

Figure 110.39: 9,297 Elements, 32,091 DOFs Model, 32 CPUs, PDD Partition/Repartition, ITR=1e-3, Imbal Tol 5%
110.4.6 Algorithm Fine-Tuning

From performance analysis results in previous sections, it has been shown that adaptive graph partitioning algorithm based on element graph can improve overall load balance for nonlinear elastic-plastic finite element calculations. Speed up has been observed on example problems. While on the other hand, we can also see as the model size increases, the efficiency of proposed PDD algorithm dropped sharply as shown in Figures 110.33 and 110.34.

So the naive implementation of PDD does not work as expected. With load balancing, one expects that the performance of PDD should not be worse than the DD case. It otherwise implies that the PDD does not bring performance gain that can completely offset its own extra load balancing operations-related overheads.

In this chapter, more detailed algorithm fine-tuning has been performed to address the problems we had in previous sections of the naive PDD implementation.

In order to improve the overall efficiency of proposed PDD algorithm, we have to consider two levels of costs when one wishes to balance the computational load among processing units. One is the data communication cost, and the other one is finite element model regeneration overhead associated with specific application problems.

Currently the adaptive graph partitioning algorithm does not consider the fact that the network communication patterns might differ much among processing nodes. The single $ITR$ value indicates the algorithmic approach of the graph partitioning algorithm, but the real communication performance has not been addressed in the implementation.

On the other hand, certain applications impose extra problem-dependent overhead to repartitioning operations. For example, whenever data communications happen, the finite element model has to be wiped off and regenerated. This is not inherent with the graph partitioning algorithm but still needs to be addressed in order to get the best performance. As observed in this chapter, model regeneration overhead increases when the finite element model becomes bigger.

In order to improve the overall performance of our application, we hope to consider both data communication and model regeneration cost and create a new strategy through which we can adaptively monitor the extra overheads to assure that load balancing operation can offset both costs.

This chapter will first investigate the effect of load balance tolerance on performance and then a new globally adaptive strategy will be proposed to handle both communication and model regeneration overhead. Speedup analysis have been done to show performance gains.
110.4.7 Fine Tuning on Load Imbalance Tolerance

If one finds out that the application-associated overhead (say, model regeneration cost) overwhelms when repartitioning happens, the most natural way to improve performance is to increase the load imbalance tolerance of the adaptive repartition routine. In this way, one hopes to increase the critical load imbalance that can trigger the balancing routine and so that the repartition counts can be reduced. As a result, model regeneration cost can do less harm to the overall performance.

This should rather viewed as a work-around and has not been effective in our application.

The tuning approach aims at improving efficiency of previous runs that failed showing speedup over static domain decomposition method. Shallow foundation model with 9,297 Elements, 32,091 DOFs has been chosen to study the effect of imbalance tolerance on parallel performance. Model setup has been the same as in previous sections.

Speedup analysis results have been shown in Figures 110.40, 110.41 and 110.42.

![Figure 110.40: Timing Data of Parallel Runs on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 20%](image)

Figure 110.40: Timing Data of Parallel Runs on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 20%
Figure 110.41: Absolute Speedup Data of Parallel Runs on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 20%
Figure 110.42: Relative Speedup of PDD over Static DD on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 20%
From the performance results, we can see that increasing load imbalance tolerance does not lead to efficiency for our application. As the number of processing units increases, the whole performance of application codes deteriorates. It is also important to note that the adaptive graph partitioning/repartitioning kernel in ParMETIS has not been capable of producing adequate partitions for finite element calculations when the load imbalance tolerance is larger than the recommended 5% Karypis et al. (2003). The application crashed with 20% imbalance tolerance for same models tested in previous sections.

The conclusion reached for the application in this chapter is that load imbalance tolerance larger than 5% has not been proved more efficient. This can also be explained in more details.

In the implementation of ParMETIS, load imbalance tolerance is one of the most important parameters in the sense that this value determines whether repartition will be switched on. The other equally significant implication of this value comes from the fact that it also establishes target load imbalance residual to be achieved after adaptive load balancing. That means for each repartition, the ParMETIS will only reduce the load imbalance to the provided tolerance.

In current implementation, the load imbalance tolerance is set to be the same for both switch-on and target values, which is not capable of bringing the best performance into our application due to the fact that aside from data redistribution cost, analysis model reconstruction is equally expensive. The dilemma is described by numerical example as shown in Table 110.5.

| Table 110.5: Observation on Load Imbalance Tolerance %5 |
|---------------------------------|---------------------------------|
| Model                           | 20,476 Elements, 68,451 DOFs    |
| CPUs                            | 32                              |
| Imbalance Before                | 7.018%                          |
| Imbalance After                 | 4.9%                            |
| Model Regeneration              | 57.2934 seconds                 |
| Total Step Time                 | 140.961 seconds                 |

We can easily see that tiny portion of data movement to balance out $7.018 - 4.9 = 2.228\%$ loads still invoked analysis model regeneration, which accounts for extra overhead that is about 40.6\% of total step time.

Because the load balance tolerance is also the target value that the repartitioning operation hopes to achieve. The implication is that after repartitioning, the load distribution among processing units is barely under this acceptable tolerance. The performance study conducted so far showed that continuous plastification can easily creates load imbalance over this tolerance so another round of repartitioning would be launched again. It greatly brings down the performance of the whole application when the
huge data redistribution overhead is taken just to overcome a tiny imbalance. This explains why changing the tolerance was not able to bring better performance in our application.

In order to improve performance while still minimizing load imbalance, we hope to maximize the efficiency of model regeneration routine in our application. This is a two-fold statement, firstly, we don’t want to blindly increase the load imbalance because it basically claims we fail our adaptive PDD algorithm by not switching on repartitioning (5% is suggested by the author of ParMETIS Karypis et al. (2003) and has been proved to be the most stable value in this chapter), secondly, with each repartitioning, we hope to achieve "perfect balance" as much as possible and in this way, the huge model regeneration cost can be offset by performance gain. What was proposed as future extension of this chapter is the idea of dual load imbalance tolerances. Load balancing triggering tolerance and the target tolerance can be defined separately. We can set higher triggering tolerance to reduce the number of repartition counts, while on the other hand a strict target tolerance can be set close to 1.0 to get better load distribution out of the balancing routine. With proposed approach, our application in this chapter will be able to fully take advantage of the repartition routines without sacrificing too much on model regenerations.

110.4.8 Globally Adaptive PDD Algorithm

One significant drawback of current implementation is that neither network communication nor model regeneration cost has been considered in element-graph-based type domain decomposition algorithm. Element graph only records computational load carried by each element. Only one $ITR$ factor characterizes algorithmic approach of the load balancing operation and this is apparently too crude for complicated network/hardware configurations. The ignorance of the repartitioning-associated overheads inherent with application codes can lead to serious performance drop of the proposed PDD algorithm as shown in Figure 110.43.

This drawback can harm the overall performance of the whole application code more seriously when the simulation is to be run on heterogeneous networks, which means we can have different network connections and nodes with varied computational power. The dilemma is, without exact monitoring of network communication and local model regeneration costs, we can easily sacrifice the performance gain by load balancing operations.

A second approach proposed in this chapter was the idea of modified Globally Adaptive PDD algorithm. The novelty comes from the fact that both data redistribution and analysis model regeneration costs will be monitored during execution. Load balancing will be triggered only when the performance gain necessarily offset the extra cost associated with the whole program. Domain graph structures will be kept intact till successful repartitioning happens. Meanwhile all elemental calculations will be timed to provide graph vertex weights. Data will be accumulated till algorithm restart happens, when all analysis
model and vertex weights will be nullified.

This improvement aims at handling network communication and any specific application-associated overheads automatically at the global level in order to remedy the drawback that the element graph repartitioning kernel currently supported by ParMETIS is not capable of directly reflecting this application level overheads. The new strategy is to automatically monitor network communication and local model regeneration timings which will be integrated to the entry of load balancing routines to act as additional triggers of the operation along with the load imbalance tolerance.

Performance study shows that PDD algorithm with the new additions significantly improve performance even when the number of processing units is large. This modification fixes the drawback shown in previous sections that the performance of PDD was beaten by static domain decomposition when the number of processors increases.

This strategy is called to be **globally adaptive** because both data communication and model regeneration costs are monitored at the application level, which tells best how the real application performs on all kinds of networks. Whatever the network/hardware configurations might be, real application runs always deliver the most accurate performance counters. This information can be applied on top of graph
partitioning algorithm as a supplement to account for the drawback that the algorithm kernel is not capable of integrating global data communication costs.

### 110.4.8.1 Implementations

We can define the global overhead associated with load balancing operation as two parts, data communication cost $T_{comm}$ and finite element model regeneration cost $T_{regen}$.

$$T_{overhead} := T_{comm} + T_{regen}$$  \hfill (110.5)

Performance counters have been setup to study both.

- $T_{comm}$
  
  Data communication patterns characterizing the network configuration can be readily measured as the program runs the initial partitioning. As described in previous sections, initial domain decomposition needs to be done to send elements over to processing nodes. This step is necessary for parallel finite element processing and it provides perfect initial estimate how the communication pattern of the application performs on specific networks. Timing routines have been added to automatically measure the communication cost. This cost is inherently changing as the network condition might vary as simulation progresses, so whenever data redistribution happens, this metric will be automatically updated to reflect the network conditions.

- $T_{regen}$
  
  Model regeneration cost basically comes from the fact that if data redistribution happens, the analysis model needs to be regenerated to reflect changes of nodes and elements inside the domain. Detailed operations include renumbering DOFs and rehandling constraints. This part of cost is application-dependent. In current implementation of PDD, efforts have been made to set up timing stop at the entry and exit of model regeneration routines to get the accurate data for the extra overhead. It is also important to note that model regeneration happens when the initial data distribution finishes, again the initial domain decomposition phase provides perfect initial estimate of the model regeneration cost on any specific hardware configurations.

Naturally, for the load balancing operations to pay off, the $T_{overhead}$ has to be offset by the performance gain $T_{gain}$. This chapter also creates a strategy to estimate the performance gain $T_{gain}$ even before the load balancing operation happens and this metric provides global control on top of the existing graph repartitioning algorithm.

As implemented in previous sections, the computational load on each element is represented by the associated vertex weight $vwgt[i]$. If the $SUM$ operation is applied on every single processing node, the
exact computational distribution among processors can be obtained as total wall clock time for each CPU as shown in Equation 110.6,

\[ T_j := \sum_{i=1}^{n} v\text{wgt}[i], \quad j = 1, 2, \ldots, np \]  

(110.6)
in which \( n \) is the number of elements on each processing domain and \( np \) is the number of CPUs.

If we define,

\[ T_{\text{sum}} := \text{sum}(T_j), \quad T_{\text{max}} := \text{max}(T_j), \quad \text{and} \quad T_{\text{min}} := \text{min}(T_j), \quad j = 1, 2, \ldots, np \]  

(110.7)

one always hope to minimize \( T_{\text{max}} \) because in parallel processing, \( T_{\text{max}} \) controls the total wall clock time. By load balancing operations, we mean to deliver evenly distributed computational loads among processors. So theoretically, the best execution time is,

\[ T_{\text{best}} := T_{\text{sum}}/np, \quad \text{and} \quad T_j \equiv T_{\text{best}}, \quad j = 1, 2, \ldots, np \]  

(110.8)

if the perfect load balance is to be achieved.

Based on definitions above, the best performance gain \( T_{\text{gain}} \) one can obtain from load balancing operations can be calculated as,

\[ T_{\text{gain}} := T_{\text{max}} - T_{\text{best}} \]  

(110.9)

Finally, the load balancing operation will be beneficial \textbf{IF AND ONLY IF}

\[ T_{\text{gain}} \geq T_{\text{overhead}} = T_{\text{comm}} + T_{\text{regen}} \]  

(110.10)

110.4.8.2 Performance Results

The newly improved design has been compared to the old design to see the effectiveness of the globally adaptive switch of PDD algorithm.
Figure 110.44: Performance of Globally Adaptive PDD on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 5%
Figure 110.45: Performance of Globally Adaptive PDD on 20,476 Elements, 68,451 DOFs Model, ITR=1e-3, Imbal Tol 5%
From Figures 110.44 and 110.45, advantage of the improved globally adaptive PDD algorithm have clearly been shown. After considering the effect of both data communication and model regeneration costs, the adaptive PPD algorithm necessarily outperforms the static Domain Decomposition approach as expected. This new design also significantly improves the overall scalability of the proposed PDD algorithm as shown in Figure 110.46 and 110.47.

Figure 110.46: Scalability Study on 4,938 Elements, 17,604 DOFs Model, ITR=1e-3, Imbal Tol 5%
Figure 110.47: Scalability Study on 9,297 Elements, 32,091 DOFs Model, ITR=1e-3, Imbal Tol 5%
110.4.9 Scalability Study on Prototype Model

The ultimate purpose of this chapter is to develop an efficient parallel simulation tool for large scale earthquake analysis on prototype SFSI system. After in-depth development-refining process conducted in previous sections, real 3-bent production models have been set up to study the parallel performance of the proposed PDD algorithm using real world earthquake ground motions.

110.4.9.1 3 Bent SFSI Finite Element Models

As described in later sections, various sizes of a 3 bent bridge SFSI system has been developed to study dynamic behaviors of the whole system in different frequency domain. These models provide perfect test cases for parallel scalability study of our proposed PDD algorithm.

Detailed model description will be presented in later chapters of this chapter and only model size and mesh pictures are shown here to indicate the range of model sizes we have covered.

Figure 110.48: Finite Element Model - 3 Bent SFSI, 56,481 DOFs, 13,220 Elements, Frequency Cutoff > 3Hz, Element Size 0.9m, Minimum $G/G_{\text{max}}$ 0.08, Maximum Shear Strain $\gamma$ 1%
Figure 110.49: Finite Element Model - 3 Bent SFSI, 484,104 DOFs, 151,264 Elements, Frequency Cutoff 10Hz, Element Size 0.3m, Minimum $G/G_{\text{max}}$ 0.08, Maximum Shear Strain $\gamma$ 1%

Figure 110.50: Finite Element Model - 3 Bent SFSI, 1,655,559 DOFs, 528,799 Elements, Frequency Cutoff 10Hz, Element Size 0.15m, Minimum $G/G_{\text{max}}$ 0.02, Maximum Shear Strain $\gamma$ 5%
110.4.9.2 Scalability Runs

The models with different detail levels have been subject to 1997 Northridge earthquake respectively for certain time steps and total wall clock time has been recorded to analyze the parallel scalability of our proposed PDD. The result is presented in Figure 110.51.

Figure 110.51: Scalability Study on 3 Bent SFSI Models, DRM Earthquake Loading, Transient Analysis, ITR=1e-3, Imbal Tol 5%, Performance Downgrade Due to Increasing Network Overhead
110.4.10 Conclusions

Through detailed performance studies as presented in previous sections, some conclusions can be drawn and future directions can be noted.

- Plastic Domain Decomposition (PDD) algorithm based on adaptive multilevel graph partitioning kernels has been shown to be effective for elastic-plastic parallel finite element calculations. PDD algorithm consistently outperforms classical Domain Decomposition method for models tested so far in this chapter as shown in Figures 110.52 and 110.54.

![Figure 110.52: Relative Performance of PDD over DD, Shallow Foundation Model, Static Loading, ITR=1e-3, Imbal Tol 5%](image-url)
There are some parameters that can be calibrated in the current implementation. As indicated by results of thorough numerical tests, $ITR=0.001$ and load imbalance tolerance $ubvec=1.05$ (5%) should be adopted and studies on our application in this chapter have shown they are adequate and able to bring performance not worse than the commonly used domain decomposition method in parallel finite element analysis.

For the parameters suggested in the chapter, we can see a general trend that the efficiency of PDD will drop as the number of processors increases. This can be explained. The implication of increasing processing units is that the subdomain problem size will decrease. It is naturally evident that the repartition load balancing won’t be able to recover the overhead by balancing off small size local calculations. The improved design of globally adaptive PDD algorithm has been implemented in this chapter and both data communication and model regeneration costs associated with graph repartitioning have been integrated into the new globally adaptive strategy. With the new design, it has also been shown that the PDD algorithm consistently outperforms classic one step domain decomposition algorithm and better scalability can be obtained as shown in Figure 110.53. It has been shown that even for large number of processors, the current implementation can always guarantee that the performance of PDD is not worse than static DD method as shown in Figure 110.54 (the repartition routine has less than 5% overhead of the total wall clock time).
Figure 110.53: Scalability of PDD, Static Loading, Shallow Foundation Model, ITR=1e-3, Imbal Tol 5%
Figure 110.54: Relative Speedup of PDD over DD, Static Loading, Shallow Foundation Model, ITR=1e-3, Imbal Tol 5%
• If the problem size is fixed, there exists an optimum number of processors that can bring the best performance of the proposed load balancing algorithm. As the number of processing units increases after this number, the efficiency of proposed algorithm drops, which is understandable because the local load imbalance is so small overall that balancing gain won’t offset the extra cost associated with repartitioning. But still the bottom line of proposed adaptive PDD algorithm is that it can run as fast as static one-step domain decomposition approach with less than 5% overhead of repartitioning routine calls. On the other hand, if the number of processing units is fixed, bigger finite element model will exhibit better performance. The conclusion is shown clearly in 3D in Figure 110.54.

• It is also worthwhile to point out that even without comparing with classical DD, PDD itself exhibits deteriorating performance as the number of processing units increases. Here the reproduction of Figure 110.53 is presented with some downside performance noted as shown in Figure 110.55.
Figure 110.55: Full Range Scalability of PDD, Static Loading, Shallow Foundation Model, ITR=1e-3, Imbal Tol 5%, Performance Downgrade Due to Increasing Network Overhead
The implication is explained as follows:

- The performance drop partly is due to the communication overhead gets bigger and bigger so parallel processing will not be able to offset the communication loss.

- It is also noted that as the number of processing units increases, the elemental level calculation drops very scalably with the number of CPUs. This is inherently advantage of the proposed PDD algorithm. PDD through domain decomposition is very scalable for local level calculations because inherently local comp is element-based. When elements are distributed, loads are spread out evenly (during initial and redistribution). So as the number of CPU increases, the equation solving becomes more expensive.

For the case of 56,481 DOFs prototype model with DRM earthquake loading, it has been observed that for sequential case (1 CPU), elemental computation takes 70% of time. As for parallel case (8 CPUs), we optimized parallel elemental computations through PDD, elemental computation only accounts for about 40%. As the number of CPU increases, parallel case (32 CPUs), the local level computation will only take less than 10% of total wall clock time. In other words, as the number of CPUs increases, PDD loses scalability because of the equation solving now dominates. As being discussed in Chapter 110.5, the parallel direct solver itself is not scalable up to large number of CPUs Demmel et al. (1999a). Parallel iterative solver is much more scalable but difficult to guarantee convergence. This is now also the most important topic in the whole scientific computing community.

For one set of fixed algorithm parameters, such as ITR and load imbalance tolerance, basic conclusion is there exists an optimal number of processors that can bring best performance and as finite element model size increases, this number increases as listed in Table 110.6.

<table>
<thead>
<tr>
<th># of DOFs</th>
<th>Speedup</th>
<th># of CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4,035</td>
<td>1.553</td>
<td>4</td>
</tr>
<tr>
<td>17,604</td>
<td>1.992</td>
<td>7</td>
</tr>
<tr>
<td>32,091</td>
<td>1.334</td>
<td>7</td>
</tr>
<tr>
<td>68,451</td>
<td>1.068</td>
<td>16</td>
</tr>
</tbody>
</table>

The second point is related to the implementation of the multilevel graph partitioning algorithm. In current implementation of ParMETIS used in this chapter, vertex weight can only be specified as an int. That means in order to get timing data from local level calculation for each element, double data
returned by MPI timing routine has to be converted to \texttt{int}. Significant digit loss can happen depending on what accuracy the system clock can carry. We can also adjust the vertex weight by amplifying the timing by scale factors in order to save effective digits. 10 millisecond has been used in this chapter to represent the effective timing digits when converting from \texttt{double} to \texttt{int}.

110.5 Application of Project-Based Iterative Methods in SFSI Problems

110.5.1 Introduction

Finite element method has been the most extensively used numerical method in computational mechanics. Equation solver is the numerical kernel of any finite element package. Gauss elimination type direct solver has dominated due to its robustness and predictability in performance.

As modern computer becomes more and more powerful, more advanced and detailed models need to be analyzed by numerical simulation. Direct solver is not the favorite choice for large scale finite element calculations because of high memory requirements and the inherent lack of parallelism of the method itself.

The motivation for presented work on iterative solvers stems from the need to expand the toolset of parallel iterative solvers for large scale simulation problems related to Earthquake-Soil-Structure interaction problems.

In this section, the effectiveness of Krylov iterative methods has been tested in solving soil-structure interaction problems. Preconditioning techniques have been introduced. Robustness of iterative solvers has been investigated on equation systems from real soil-structure interaction problems. Several popular parallel algorithms and tools have been collected and implemented on PETSc platform to solve the SFSI problems. Performance study has been carried out using IA64 super computers at San Diego Supercomputing Center. A complete implementation has been developed within our computational system, within MOSS libraries, with extensive use of ParMETIS, and other material and numerical libraries.

110.5.2 Projection-Based Iterative Methods

Projection techniques are defined as methods to find approximate solutions $\hat{x}$ for $Ax = b$ ($A \in \mathbb{R}^{n \times n}$) in a subspace $\mathcal{W}$ of dimension $m$. Then in order to determine $\hat{x}$, we need $m$ independent conditions. One way to obtain these is by requiring the residual $b - A\hat{x}$ is orthogonal to a subspace $\mathcal{V}$ of dimension $m$. 
\[ \hat{x} \in W, b - A\hat{x} \perp V \]  \hspace{1cm} (110.11)

The conditions shown in Equation 110.11 are known as Petrov-Galerkin conditions (Bai, 2007).

There are two key questions to answer if one wants to use projection techniques in solving large scale linear systems. Different answers lead to many variants of the projection method.

- **Choice of Subspaces**
  Krylov subspaces have been the favorite of most researchers and a large family of methods have been developed based on Krylov subspaces. Typically people choose either \( V = W \) or \( V = AW \) with \( V \) and \( W \) both Krylov subspaces.

- **Enforcement of Petrov-Galerkin Conditions**
  Arnoldi’s procedure and Lanczos algorithm are two choices for building orthogonal or biorthogonal sequence to enforce the projection conditions.

The iterative methods discussed in this section are generally split into two categories, one based on Arnoldi’s procedure and the other on Lanczos biorthogonalization. The most popular for the first family are Conjugate Gradient and General Minimum Residual methods, while Bi-Conjugate Gradient and Quasi-Minimum Residual methods represent the Lanczos family.

### 110.5.2.1 Conjugate Gradient Algorithm

The conjugate gradient (CG) algorithm is one of the best known iterative techniques for solving sparse symmetric positive definite (SPD) linear systems. This method is a realization of an orthogonal projection technique onto the Krylov subspace \( K_m(A, r_0) \), where \( r_0 \) is the initial residual. Because \( A \) is symmetry, some simplifications resulting from the three-term Lanczos recurrence will lead to more elegant algorithms (Demmel, 1997).

**ALGORITHM CG** (Saad, 2003)

1. Compute \( r_0 := b - Ax_0, p_0 := r_0 \)
2. For \( j = 0, 1, \ldots \), until convergence, Do
   3. \( \alpha_j := (r_j, r_j) / (Ap_j, p_j) \)
   4. \( x_{j+1} := x_j + \alpha_j p_j \)
   5. \( r_{j+1} := r_j - \alpha_j Ap_j \)
   6. \( \beta_j := (r_{j+1}, r_{j+1}) / (r_j, r_j) \)
   7. \( p_{j+1} := r_{j+1} + \beta_j p_j \)
8. EndDo
• **Applicability**
  Matrix $A$ is SPD.

• **Subspaces**
  Choose $\mathcal{W} = \mathcal{V} = \mathcal{K}_m(A, r_0)$, in which initial residual $r_0 = b - Ax_0$.

• **Symmetric Lanczos Procedure**
  This procedure can be viewed as a simplification of the Arnoldi’s procedure when $A$ is symmetric. Great three-term Lanczos recurrence is discovered when the symmetry of $A$ is considered (Demmel, 1997).

• **Optimality**
  If $A$ is SPD and one chooses $\mathcal{W} = \mathcal{V}$, enforcing Petrov-Galerkin conditions minimizes the $A$-norm of the error over all vectors $x \in \mathcal{W}$, i.e., $\hat{x}$ solves the problem,

$$
\min_{x \in \mathcal{W}} \|x - x^*\|_A, x^* = A^{-1}b
$$

(110.12)

From the lemma above, one can derive global minimization property of the Conjugate Gradient method. The vector $x_k$ in the Conjugate Gradient method solves the minimization problem

$$
\min_x \phi(x) = \frac{1}{2} \|x - x^*\|_A^2, x - x_0 \in \mathcal{K}_k(A, r_0)
$$

(110.13)

• **Convergence**
  In exact arithmetic, the Conjugate Gradient method will produce the exact solution to the linear system $Ax = b$ in at most $n$ steps and it owns the superlinear convergence rate. The behavior of Conjugate Gradient algorithm in finite precision is much more complex. Due to rounding errors, orthogonality is lost quickly and finite termination does not hold anymore. What is more meaningful in application problems would be to use CG method for solving large, sparse, well-conditioned linear systems in far fewer than $n$ iterations.

### 110.5.2.2 GMRES

The Generalized Minimum Residual method is able to deal with more general type of matrices.
ALGORITHM GMRES (Saad, 2003)

1. Compute \( r_0 := b - Ax_0, \beta := \|r_0\|_2, \) and \( v_1 := r_0/\beta \)
2. For \( j = 1, 2, \ldots, m, \) Do
   3. Compute \( \omega_j := Av_j \)
   4. For \( i = 1, \ldots, j, \) Do
      5. \( h_{ij} := (\omega_j, v_i) \)
      6. \( \omega_j := \omega_j - h_{ij}v_i \)
   7. EndDo
   8. \( h_{j+1,j} = \|\omega_j\|_2. \) If \( h_{j+1,j} = 0 \) set \( m := j \) and go to 11
   9. \( v_{j+1} = \omega_j/h_{j+1,j} \)
10. EndDo
11. Define the \((m + 1) \times m\) Hessenberg matrix \( \bar{H}_m = \{h_{ij}\}_{1 \leq i \leq m+1, 1 \leq j \leq m} \)
12. Compute \( y_m, \) the minimizer of \( \|\beta_i e_1 - \bar{H}_m y\|_2, \) and \( x_m = x_0 + V_m y_m \)

- **Applicability**
  Matrix \( A \) is nonsingular.

- **Subspaces**
  Choose \( \mathcal{W} = \mathcal{K}_m(A, r_0) \) and \( \mathcal{V} = AW = AK_m(A, r_0), \) in which initial residual \( r_0 = b - Ax_0. \)

- **Arnoldi’s Procedure**
  Classic Arnoldi’s procedure (modified Gram-Schmidt) is followed in GMRES (Bai, 2007).

- **Optimality**
  If one chooses \( \mathcal{V} = AW, \) enforcing Petrov-Galerkin conditions solves the least square problem
  \[
  \|b - A\tilde{x}\|_2 = \min_{x \in \mathcal{W}} \|b - Ax\|_2 
  \] (110.14)

- **Convergence**
  It has been shown that in exact arithmetic, GMRES can not breakdown and will give exact solutions in at most \( n \) steps. In practice, the maximum steps GMRES can run depends on the memory due to the fact it needs to store all Arnoldi vectors. Restarting schemes have been proposed for a fixed \( m, \) which is denoted by \( \text{GMRES}(m). \) Typical value for \( m \) can be \( m \in [5, 20]. \) GMRES\((m)\) can not breakdown in exact arithmetic before the exact solution has been reached. But it may never converge for \( m < n \) (Bai, 2007).
110.5.2.3 BiCGStab and QMR

These two methods are based on nonsymmetric Lanczos procedure, which is quite different from Arnoldi’s in the sense that it formulates biorthogonal instead of orthogonal sequence. They are counterparts of CG and GMRES method, which follows similar derivation procedure except the Lanczos biorthogonalization is used instead of Arnoldi’s procedure (Bai, 2007).

110.5.3 Preconditioning Techniques

Lack of robustness is a widely recognized weakness of iterative solvers relative to direct solvers. Using preconditioning techniques can greatly improve the efficiency and robustness of iterative methods. Preconditioning is simply a means of transforming the original linear system into one with the same solution but easier to solve with an iterative solver. Generally speaking, the reliability of iterative techniques, when dealing with various applications, depends much more on the quality of the preconditioner than on the particular Krylov subspace accelerator used.

The first step in preconditioning is to find a preconditioning matrix $M$. The matrix $M$ can be defined in many different ways but there are a few minimal requirements the $M$ is supposed to satisfy (Benzi, 2002).

1. From practical point of view, the most important requirement of $M$ is that it should be inexpensive to solve linear system $Mx = b$. This is because the preconditioned algorithm will all require a linear system solution with the matrix $M$ at each step.

2. The matrix $M$ should be somehow close to $A$ and it should not be singular. We can see that actually most powerful preconditioners are constructed directly from $A$.

3. The preconditioned $M^{-1}A$ should be well-conditioned or has very few extreme eigenvalues thus $M$ can accelerate convergence dramatically.

Once a preconditioner $M$ is available, there are three ways to apply it.

1. Left Preconditioning

   $$M^{-1}Ax = M^{-1}b$$

2. Right Preconditioning

   $$AM^{-1}u = b, x \equiv M^{-1}u$$
3. Split Preconditioning

It is a very common situation that $M$ is available in factored form $M = M_L M_R$, in which, typically, $M_L$ and $M_R$ are triangular matrices. Then the preconditioning can be split,

$$M_L^{-1} A M_R^{-1} u = b, x \equiv M_R^{-1} u \quad (110.17)$$

It is imperative to preserve symmetry when the original matrix $A$ is symmetric, so the split preconditioner seems mandatory in this case.

Consider that a matrix $A$ that is symmetric and positive definite and assume that a preconditioner $M$ is available. The preconditioner $M$ is a matrix that approximates $A$ in some yet-undefined sense. We normally require that the $M$ is also symmetric positive definite.

In order to preserve the nice SPD property, in the case when $M$ is available in the form of an incomplete Cholesky factorization, $M = LL^T$, people can simply just use the split preconditioning, which yields the SPD matrix

$$L^{-1} A L^{-T} u = L^{-1} b, x \equiv L^{-T} u \quad (110.18)$$

However, it is not necessary to split the preconditioner in this manner in order to preserve symmetry. Observe that $M^{-1} A$ is self-adjoint for the $M$ inner product

$$(x, y)_M \equiv (M x, y) = (x, M y) \quad (110.19)$$

since

$$(M^{-1} A x, y)_M = (A x, y) = (x, A y) = (x, M(M^{-1} A)y) = (x, M^{-1} A y)_M \quad (110.20)$$

Therefore, an alternative is to replace the usual Euclidean inner product in the CG algorithm with the $M$ inner product (Saad, 2003).

If the CG algorithm is rewritten for this new inner product, denoting by $r_j = b - A x_j$ the original residual and by $z_j = M^{-1} r_j$ the residual for the preconditioned system, the following sequence of operations is obtained, ignoring the initial step:

1. $\alpha_j := (z_j, z_j)_M/(M^{-1} A p_j, p_j)_M$,
2. $x_{j+1} := x_j + \alpha_j p_j$,
3. $r_{j+1} := r_j - \alpha_j A p_j$ and $z_{j+1} := M^{-1} r_{j+1}$,
4. $\beta_j := (z_{j+1}, z_{j+1})_M/(z_j, z_j)_M$,
5. $p_{j+1} := z_{j+1} + \beta_j p_j$.

Since $(z_j, z_j)_M = (r_j, z_j)$ and $(M^{-1}Ap_j, p_j)_M = (Ap_j, p_j)$, the $M$ inner products do not have to be computed explicitly. With this observation, the following algorithm is obtained.

**Algorithm Preconditioned CG (Saad, 2003)**

1. Compute $r_0 := b - Ax_0$, $z_0 := M^{-1}r_0$, $p_0 := z_0$
2. For $j = 0, 1, \ldots, \text{until convergence}$, Do
3. \[ \alpha_j := (r_j, z_j)/(Ap_j, p_j) \]
4. \[ x_{j+1} := x_j + \alpha_j p_j \]
5. \[ r_{j+1} := r_j - \alpha_j A p_j \]
6. \[ z_{j+1} := M^{-1}r_{j+1} \]
7. \[ \beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j) \]
8. \[ p_{j+1} := z_{j+1} + \beta_j p_j \]
9. EndDo

110.5.4 Preconditioners

Finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science. Theoretical results are rare and some methods work surprisingly well, often despite expectations. As it is mentioned before, the preconditioner $M$ is always close to $A$ in some undefined-yet sense. Some popular preconditioners will be introduced in this section.

110.5.4.1 Jacobi Preconditioner

This might be the simplest preconditioner people can think of. If $A$ has widely varying diagonal entries, we may just use diagonal preconditioner $M = \text{diag}(a_{11}, \cdots, a_{nn})$. One can show that among all possible diagonal preconditioners, this choice reduces the condition number of $M^{-1}A$ to within a factor of $n$ of its minimum value.

110.5.4.2 Incomplete Cholesky Preconditioner

Another simple way of defining a preconditioner that is close to $A$ is to perform an incomplete Cholesky factorization of $A$. Incomplete factorization formulates an approximation of $A \approx \hat{L}\hat{L}^T$, but with less or no fill-ins relative to the complete factorization $A = LL^T$ (Demmel, 1997).
ALGORITHM Incomplete Cholesky Factorization (Saad, 2003)

1. For $j = 1, 2, \ldots, n$, Do
2. \[ l_{jj} := \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2} \]
3. For $i = j + 1, \ldots, n$, Do
4. \[ l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj} \]
5. Apply dropping rule to $l_{ij}$
6. EndDo
7. EndDo

There are many ways to control the number of fill-ins in IC factorization. No fill-in version of incomplete Cholesky factorization IC(0) is rather easy and inexpensive to compute. On the other hand, it often leads to a very crude approximation of $A$, which may result in the Krylov subspace accelerator requiring too many iterations to converge. To remedy this, several alternative incomplete factorizations have been developed by researchers by allowing more fill-in in $L$, such as incomplete Cholesky factorization with dropping threshold IC($\epsilon$). In general, more accurate IC factorizations require fewer iterations to converge, but the preprocessing cost to compute the factors is higher.

110.5.4.3 Robust Incomplete Factorization

Incomplete factorization preconditioners are quite effective for many application problems but special care must be taken in order to avoid breakdowns due to the occurrence of non-positive pivots during the incomplete factorization process.

The existence of an incomplete factorization $A \approx \hat{L}\hat{L}^T$ has been established for certain classes of matrices. For the class of $M$-matrices, the existence of incomplete Cholesky factorization was proved for arbitrary choices of the sparsity pattern (Meijerink and van der Vorst, 1977). The existence result was extended shortly thereafter to a somewhat larger class (that of $H$-matrices with positive diagonal entries) (Manteuffel, 1980; Varga et al., 1980; Robert, 1982). Benzi and Tůma (2003) presents reviews on the topic of searching for robust incomplete factorization algorithms and an robust algorithm based on $A$-Orthogonalization has been proposed.

In order to construct triangular factorization of $A$, the well-known is not the only choice. Benzi and Tůma (2003) shows how the factorization $A = LDL^T$ (root-free factorization) can be obtained by means of an $A$-orthogonalization process applied to the unit basis vectors $e_1, e_2, \ldots, e_n$. This is simply the Gram-Schmidt process with respect to the inner product generated by the SPD matrix $A$. This idea is not new and as a matter of fact, it was originally proposed at as early as 1940’s in Fox et al. (1948). It has been observed in Hestenes and Stiefel (1952) that $A$-orthogonalization of the unit basis
vectors is closely related to Gaussian elimination but this algorithm costs twice as much as the Cholesky factorization in the dense case.

**Factored Approximate Inverse Preconditioner** In reference Benzi et al. (1996) $A$-orthogonalization has been exploited to construct factored sparse approximate inverse preconditioners noting the fact that $A$-orthogonalization also produces the inverse factorization $A^{-1} = ZD^{-1}Z^T$ (with $Z$ unit upper triangular and $D$ diagonal). Because the $A$-orthogonalization, even when performed incompletely, is not subject to pivot breakdowns, these preconditioners are reliable (Benzi et al., 2000). However, they are often less effective than incomplete Cholesky preconditioning at reducing the number of PCG iterations and their main interest stems from the fact that the preconditioning operation can be applied easily in parallel because triangular solve is not necessary in approximate inverse preconditioning.

Reference Benzi and Tůma (2003) investigates the use of $A$-orthogonalization as a way to compute an incomplete factorization of $A$ rather than $A^{-1}$ thus a reliable preconditioning algorithm can be developed. The basic $A$-orthogonalization procedure can be written as follows (Benzi et al., 2000).

**ALGORITHM Incomplete Factored Approximate Inverse (Benzi et al., 1996)**

1. Let $z_i^{0} = e_i$, for $i = 1, 2, \cdots , n$
2. For $i = 1, 2, \cdots , n$, Do
   3. For $j = i, i+1, \cdots , n$, Do
      4. $p_j^{(i-1)} := a_i^T z_j^{(i-1)}$
   5. EndDo
6. For $j = i+1, \cdots , n$, Do
   7. $z_j^{(i)} := z_j^{(i-1)} - \left( \frac{p_j^{(i-1)}}{p_i^{(i-1)}} \right) p_i^{(i-1)}$
   8. Apply dropping to $z_j^{(i)}$
   9. EndDo
10. EndDo
11. Let $z_i := z_i^{(i-1)}$ and $p_i := p_i^{(i-1)}$, for $i = 1, 2, \cdots , n$.
12. Return $Z = [z_1, z_2, \cdots , z_n]$ and $D = diag(p_1, p_2, \cdots , p_n)$.

The basic algorithm described above can suffer a breakdown when a negative or zero value of a pivot $p_i$. When no dropping is applied, $p_i = z_i^T A z_i > 0$. The incomplete procedure is well defined, i.e., no breakdown can occur, if $A$ is an $H$-matrix (in the absence of round-off). In the general case, breakdowns can occur. Breakdowns have a crippling effect on the quality of the preconditioner. A negative $p_i$ would result in an approximate inverse which is not positive definite; a zero pivot would force termination of
the procedure, since step (7) cannot be carried out.

The way proposed to avoid non-positive pivots is simply to recall that in the exact $A$-orthogonalization process, the $p_i$’s are the diagonal entries of matrix $D$ which satisfies the matrix equation

$$Z^T A Z = D$$

hence for $1 < i < n$

$$p_i = z_i^T A z_i > 0$$

since $A$ is SPD and $z_i \neq 0$. In the exact process, the following equality holds

$$p_i = z_i^T A z_i = a_i^T z_i \quad \text{and} \quad p_j = z_j^T A z_j = a_j^T z_j$$

Clearly it is more economical to compute the pivots using just inner product $a_i^T z_i$ rather than the middle expression involving matrix-vector multiply. However, because of dropping and the resulting loss of $A$-orthogonality in the approximate $\tilde{z}$-vectors, such identities no longer hold in the inexact process and for some matrices one can have

$$a_i^T \tilde{z}_i \ll \tilde{z}_i^T A \tilde{z}_i$$

The robust algorithm requires that the incomplete pivots $\bar{p}_i$’s be computed using the quadratic form $\tilde{z}_i^T A \tilde{z}_i$ throughout the AINV process, for $i = 1, 2, \cdots, n$.

**ALGORITHM Stabilized Incomplete Approximate Inverse (Benzi et al., 2000)**

1. Let $z_i^{(0)} = e_i$, for $i = 1, 2, \cdots, n$
2. For $i = 1, 2, \cdots, n$, Do
3. $v_i := A z_i^{(i-1)}$
4. For $j = i, i+1, \cdots, n$, Do
5. $p_{ij}^{(i-1)} := v_i^T z_j^{(i-1)}$
6. EndDo
7. For $j = i+1, \cdots, n$, Do
8. $z_j^{(i)} := z_j^{(i-1)} - \left( \frac{p_{ij}^{(i-1)}}{p_i^{(i-1)}} \right)$
9. Apply dropping to $z_j^{(i)}$
10. EndDo
11. EndDo
12. Let $z_i := z_i^{(i-1)}$ and $p_i := p_i^{(i-1)}$, for $i = 1, 2, \cdots, n$.
13. Return $Z = [z_1, z_2, \cdots, z_n]$ and $D = diag(p_1, p_2, \cdots, p_n)$. 

Obviously, the robust (referred to as SAINV) and plain algorithm are mathematically equivalent. However, the incomplete process obtained by dropping in the $z$-vectors in step (9) of the robust algorithm leads to a reliable approximate inverse. This algorithm, in exact arithmetic, is applicable to any SPD matrix without breakdowns. The computational cost of SAINV is higher than basic AINV and special care has to be taken to do sparse-sparse matrix-vector multiply.

**Incomplete Factorization by SAINV**  Consider now the exact algorithm (with no dropping) and write $A = LDL^T$ with $L$ unit lower triangular and $D$ diagonal. Observe that $L$ in the $LDL^T$ factorization of $A$ and the inverse factor satisfy
\[
AZ = LD \quad \text{or} \quad L = AZD^{-1}
\]  where $D$ is the diagonal matrix containing the pivots. This easily follows from
\[
Z^T AZ = D \quad \text{and} \quad Z^T = L^{-1}
\]  If we recall that pivot $d_j = p_j = z_j^T A z_j = \langle Az_j, z_j \rangle$, then by equating corresponding entries of $AZD^{-1}$ and $L = [l_{ij}]$ we find that (Benzi and Tůma, 2003; Bollhöfer and Saad, 2001)
\[
l_{ij} = \frac{\langle Az_j, z_i \rangle}{\langle Az_i, z_j \rangle} \quad i \geq j
\]  Hence, the $L$ factor of $A$ can be obtained as a by-product of the $A$-orthogonal-ization, at no extra cost. In the implementation of SAINV, the quantities $l_{ij}$ in Equation 110.27 are the multipliers that are used in updating the columns of $Z$. Once the update is computed, they are no longer needed and are discarded. To obtain an incomplete factorization of $A$, we do just the opposite; we save the multipliers $l_{ij}$, and discard the column vectors $z_j$ as soon as they have been computed and operated with. Hence, the incomplete $L$ factor is computed by columns; these columns can be stored in place of the $z_j$ vectors, with minimal modifications to the code. Here, we are assuming that the right-looking form of SAINV is being used. If the left-looking one is being used, then $L$ would be computed by rows. Please refer to Benzi and Tůma (2003) for more implementation details.

**110.5.5 Numerical Experiments**

Matrices from soil-structure interaction finite element analysis have been extracted from simulation system to study the performance of different preconditioning techniques on PCG method. The prototype of soil structure model has been shown in Figures 110.56 and 110.57. In order to introduce both of the nonlinear theories for soil and structures, we use continuum elements to model the soil and beam
elements for the structures. Matrices from static pushover analysis and dynamic ground motion analysis have been collected for this research.

Figure 110.56: Finite Element Mesh of Soil-Structure Interaction Model
Figure 110.57: Finite Element Mesh of Soil-Structure Interaction Model
Figure 110.58: Matrices $N = 3336$ (Continuum FEM)

Figure 110.59: Matrices $N = 5373$ (Continuum FEM)
Figure 110.60: Matrices $N = 33081$ (Continuum FEM)

Figure 110.61: Matrices $N = 8842$ (Soil-Beam Static FEM)
Table 110.7: Matrices in FEM Models

<table>
<thead>
<tr>
<th>Continuum Model (Static)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix</strong></td>
</tr>
<tr>
<td>m1188</td>
</tr>
<tr>
<td>m1968</td>
</tr>
<tr>
<td>m11952</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Soil-Beam Model (Static and Dynamic)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix</strong></td>
</tr>
<tr>
<td>SoilBeam</td>
</tr>
<tr>
<td>SoilBeamDyn</td>
</tr>
</tbody>
</table>

Figure 110.62: Matrices $N = 8842$ (Soil-Beam Dynamic FEM)
SPD matrices have been studied using Conjugate Gradient method with or without preconditioning. Performance has been summarized in Table 110.8.

Table 110.8: Performance of CG and PCG Method (Continuum FEM)

| Preconditioner | # Iter | Pre Time(s) | Iter Time(s) | Total Time(s) | Density
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>4376</td>
<td>-</td>
<td>54.82</td>
<td>54.82</td>
<td>-</td>
</tr>
<tr>
<td>Jacobi</td>
<td>1612</td>
<td>0.01</td>
<td>20.18</td>
<td>20.19</td>
<td>-</td>
</tr>
<tr>
<td>IC(0)</td>
<td>413</td>
<td>2.19</td>
<td>11.07</td>
<td>13.26</td>
<td>1.00</td>
</tr>
<tr>
<td>IC(1e-6)</td>
<td>5</td>
<td>5.90</td>
<td>0.47</td>
<td>6.37</td>
<td>5.88</td>
</tr>
<tr>
<td>RIF2(1e-2)</td>
<td>571</td>
<td>9.37</td>
<td>14.94</td>
<td>24.31</td>
<td>0.94</td>
</tr>
<tr>
<td>RIF3(1e-2)</td>
<td>541</td>
<td>6.80</td>
<td>14.13</td>
<td>20.93</td>
<td>0.94</td>
</tr>
</tbody>
</table>

| Preconditioner | # Iter | Pre Time(s) | Iter Time(s) | Total Time(s) | Density
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>4941</td>
<td>-</td>
<td>103.78</td>
<td>103.78</td>
<td>-</td>
</tr>
<tr>
<td>Jacobi</td>
<td>1711</td>
<td>0.01</td>
<td>36.61</td>
<td>36.62</td>
<td>-</td>
</tr>
<tr>
<td>IC(0)</td>
<td>437</td>
<td>6.5</td>
<td>20.38</td>
<td>26.88</td>
<td>1.00</td>
</tr>
<tr>
<td>IC(1e-6)</td>
<td>6</td>
<td>19.81</td>
<td>1.3</td>
<td>21.11</td>
<td>8.10</td>
</tr>
<tr>
<td>RIF2(1e-2)</td>
<td>599</td>
<td>25.71</td>
<td>26.55</td>
<td>52.26</td>
<td>0.96</td>
</tr>
<tr>
<td>RIF3(1e-2)</td>
<td>566</td>
<td>21.31</td>
<td>25.23</td>
<td>46.54</td>
<td>0.96</td>
</tr>
</tbody>
</table>

| Preconditioner | # Iter | Pre Time(s) | Iter Time(s) | Total Time(s) | Density
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>6754</td>
<td>-</td>
<td>952.53</td>
<td>952.53</td>
<td>-</td>
</tr>
<tr>
<td>Jacobi</td>
<td>2109</td>
<td>0.03</td>
<td>308.46</td>
<td>308.49</td>
<td>-</td>
</tr>
<tr>
<td>IC(0)</td>
<td>565</td>
<td>273.83</td>
<td>173.05</td>
<td>446.88</td>
<td>1.00</td>
</tr>
<tr>
<td>IC(1e-6)(^3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIF2(1e-2)</td>
<td>694</td>
<td>1172.7</td>
<td>211.88</td>
<td>1384.58</td>
<td>0.99</td>
</tr>
<tr>
<td>RIF3(1e-2)</td>
<td>664</td>
<td>1245.4</td>
<td>202.67</td>
<td>1448.07</td>
<td>0.99</td>
</tr>
</tbody>
</table>

\(^2\) Density is defined as the number of non-zeros of the incomplete factor divided by the number of non-zeros in the lower triangular part of $A$.

\(^3\) Could not continue because memory requirement larger than 1.4GB.
Figure 110.63: Convergence of CG and PCG Method (3336 DOFs Model)

Figure 110.64: Convergence of CG and PCG Method (5373 DOFs Model)
### Table 110.9: Performance of CG and PCG Method (Soil-Beam FEM)

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th># Iter</th>
<th>Pre Time(s)</th>
<th>Iter Time(s)</th>
<th>Total Time(s)</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>3274</td>
<td></td>
<td></td>
<td>102.5</td>
<td>102.5</td>
</tr>
<tr>
<td>Jacobi</td>
<td>1687</td>
<td>0.01</td>
<td>54.56</td>
<td>54.57</td>
<td>-</td>
</tr>
<tr>
<td>IC(0)</td>
<td>26</td>
<td>15.77</td>
<td>1.95</td>
<td>17.72</td>
<td>1.00</td>
</tr>
<tr>
<td>IC(1e-6)</td>
<td>6</td>
<td>110.17</td>
<td>2.79</td>
<td>112.96</td>
<td>15.11</td>
</tr>
<tr>
<td>RIF2(1e-6)⁵</td>
<td>23</td>
<td>3364.8</td>
<td>3.44</td>
<td>3368.24</td>
<td>4.32</td>
</tr>
<tr>
<td>RIF3(1e-6)⁵</td>
<td>31</td>
<td>34541</td>
<td>9.26</td>
<td>34550.26</td>
<td>16.37</td>
</tr>
</tbody>
</table>

### 8842 DOFs Soil-Beam FEM (Static)

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th># Iter</th>
<th>Pre Time(s)</th>
<th>Iter Time(s)</th>
<th>Total Time(s)</th>
<th>Density⁴</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>3276</td>
<td></td>
<td></td>
<td>136.7</td>
<td>136.7</td>
</tr>
<tr>
<td>Jacobi</td>
<td>MaxIt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IC(0)</td>
<td>MaxIt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IC(1e-6)</td>
<td>MaxIt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIF2(1e-2)</td>
<td>MaxIt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIF3(1e-2)</td>
<td>MaxIt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⁴Density is defined as the number of non-zeros of the incomplete factor divided by the number of non-zeros in the lower triangular part of $A$.

⁵Iteration with tolerance 1e-2 failed to converge.

### 110.5.6 Conclusion and Future Work

1. For the soil-structure interaction problems investigated in this section, Conjugate Gradient method works fine and the convergence is acceptable for most cases.

2. Incomplete Cholesky factorization preconditioner has been shown to be very powerful in static pushover problems.

3. Dynamic problems formulated by Newmark integration scheme have not been extensively tested. But according to the data available so far, neither IC nor RIF preconditioners performed well and further testing is necessary to reach a more persuasive conclusion. The difficulty in dynamic analysis results from the fact that consistent mass and damping matrices used in continuum finite element formulations significantly degrade the conditioning number of the final system. This situation deteriorates when penalty handler is used to apply multiple point constraints, which introduces huge off-diagonal numbers to stiffness, mass and damping matrices (Cook et al., 2002).

4. Robust incomplete factorization preconditioning based on A-orthogonalization has not been shown competitive with IC preconditioners in this research. It is also worth noting that all timings are
taken in MATLAB. There are much more improvement can be achieved with a carefully coded FORTRAN program.

5. Static analysis has been extensively studied and it can be safely concluded that IC(0) and Jacobi preconditioners are good choices for the nonlinear soil-beam interaction simulations.

6. Dynamic analysis has also been studied but more work is needed to draw any detailed conclusion. Generally speaking, one should be alert if iterative solver is to be used for dynamic analysis. This partially comes from the fact that mass and damping matrices undoubtedly alter the structures of the coefficient matrix. This situation becomes more complicated if penalty handler is used to introduce off-diagonal numbers when handling multi-point constraints. So direct solver would be a more stable option for solving dynamic equations.

110.6 Performance Study on Parallel Direct/Iterative Solving in SFSI

The motivation of this section is to introduce a robust and efficient parallel equation solver into our parallel finite element analysis framework. Aside from sparsity, which has been well known as the
Figure 110.66: Convergence of CG and PCG Method (Soil-Beam Static Model)

Figure 110.67: Convergence of CG and PCG Method (Soil-Beam Dynamic Model)
result of compact support that is inherent with finite element method, there exist some other special considerations that make the equation solving in finite element simulation a more involved problem.

In nonlinear finite element simulations, handling of constraints significantly affects the condition number of assembled equation systems. In SFSI simulations, multiple-point constraint is necessary to enforce the connection between soil and pile elements. In this research, penalty handler has been adopted to impose multiple point constraints on the assembled equation systems. Transformation and Lagrange multipliers are among those popular methods as well (Belytschko et al., 2000; Cook et al., 2002). The method of Lagrange multipliers adds extra constraints to the system and the resulted coefficient matrix will lose symmetric positive definiteness. Transformation is favorable especially in the sense that it reduces the order of the equation systems by condensing out slave/constrained DOFs. But the transformation is the most difficult to code and the situation of one single main/retained node with multiple follower/constrained nodes further complicates the problem.

Penalty method is chosen in this research due to the fact that it well preserves the symmetric positive definiteness of the system if the nice property is observed. Another consideration comes from the easiness with which the penalty methods can handle the single main/retained multiple follower/constrained situations. This is proven to be extremely valuable when data redistribution is required in adaptive parallel processing because the DOF Graph object can be clearly tracked during partition and repartition phases.

The incapability of handling constraints accurately has been long known as the weakness of penalty method. The choice of the key penalty number seems arbitrary and largely depends on experience. The dilemma is with larger penalty number, the system can handle constraints more accurately while the coefficient matrix can become very ill-conditioned. This can lead to serious convergence problem for iterative solvers.

The majority of coefficient matrices resulted from finite element analysis are inherently symmetric positive definite, for which lots of numerical algorithms have been proposed and solving SPD, symmetric or closely symmetric systems has been relatively maturer than more common unsymmetric cases. Unfortunately, in geotechnical finite element simulations, unassociated constitutive models lead to unsymmetric stiffness matrices (Jeremić, 2004). More general parallel solvers must be coded to solve the problem.

In this section, both iterative and direct solvers are coded using the consistent PETSc interface (Balay et al., 2001, 2004, 1997). Popular direct solvers for general unsymmetric systems such as MUMPS, SPARLES, SuperLU, PLAPACK have been introduced and performance study has been carried out to investigate the efficiency of different solvers on large scale SFSI simulations with penalty-handled unsymmetric equation systems. GMRES is always the first choice of iterative method when general unsymmetric systems are concerned. Preconditioning techniques have been thoroughly studied in this
research to explore possible advantage of preconditioned iterative solver over direct solving. Jacobi, incomplete LU decomposition and approximate inverse preconditioners represent the most popular choices for Krylov methods and they are chosen in this performance survey.

All numerical algorithms have been implemented through interface of PETSc, which provides a consistent platform on which implementation issues can be avoided to expose individual algorithmic performance.

110.6.1 Parallel Sparse Direct Equation Solvers

The methods that we consider for the solution of sparse linear equations can be grouped into four main categories: general techniques, frontal methods, multifrontal approaches and supernodal algorithms (Dongarra et al., 1996).

110.6.1.1 General Techniques – SPOOLES

The so-called general approach can be viewed as parallel versions of sparse LU decomposition. Special cares must be taken to handle the sparse data structures. Sparsity ordering is crucial in parallel sparse equation solving in order to reduce fill-in and discover large-grain parallelism (Demmel et al., 1993).

Freely available package SPOOLES provides minimum degree (multiple external minimum degree (Liu, 1985)), generalized nested dissection and multisection ordering schemes for matrix sparsity ordering. Fundamental supernode tree built on top of vertex elimination tree is used to explore granularity in parallel (Ashcraft, 1999; Ashcraft et al., 1999).

110.6.1.2 Frontal and Multifrontal Methods – MUMPS

Frontal methods have their origins in the solution of finite element problems from structural analysis. The usual way to describe the frontal method is to view its application to finite element problems where the matrix $A$ is expected as a sum of contributions from the elements of a finite element discretization (Dongarra et al., 1996). That is,

$$A = \sum_{l=1}^{m} A^{[l]},$$

(110.28)

where $A^{[l]}$ is nonzero only in those rows and columns that correspond to variables in the $l$th element. If $a_{ij}$ and $a_{ij}^{[l]}$ denote the $(i,j)$th entry of $A$ and $A^{[l]}$, respectively, the basic assembly operation when forming $A$ is of the form

$$a_{ij} \leftarrow a_{ij} + a_{ij}^{[l]}.$$

(110.29)
It is evident that the basic operation in Gaussian elimination

\[ a_{ij} \leftarrow a_{ij} + a_{ip}[a_{pp}]^{-1}a_{pj}. \]  

(110.30)

may be performed as soon as all the the terms in the triple product 110.30 are fully summed (that is, are involved in no more sums of the form 110.29). The assembly and Gaussian elimination processes can therefore be interleaved and the matrix \( A \) is never assembled explicitly. This allows all intermediate working to be performed in a dense matrix, termed frontal matrix, whose rows and columns correspond to variables that have not yet been eliminated but occur in at least one of the elements that have been assembled.

For general problems other than finite element, the rows of \( A \) (equations) are added into the frontal matrix one at a time. A variable is regarded as fully summed whenever the equation in which it last appears is assembled. The frontal matrix will, in this case, be rectangular.

The idea of multifrontal method is to couple a sparsity ordering with the efficiency of a frontal matrix kernel so allowing good exploitation of high performance computers. The basic approach is to develop separate fronts simultaneously which can be chosen using a sparsity preserving ordering such as minimum degree.

Elimination tree, again is the most important notion in the factorization process and also utilized to discover the potential of parallelism. An elimination tree defines the a precedence order within the factorization. The factorization commences at the leaves of of the tree and data is passed towards the root along the edges in the tree. To complete the work associated with a node, all the data must have been obtained from the children of the node, otherwise work at different nodes is independent.

Freely available package MUMPS (MUltifrontal Massively Parallel sparse direct Solver) has been used in this research to investigate the performance of multifrontal methods (http://graal.ens-lyon.fr/MUMPS/, 2006).

MUMPS is a package for solving systems of linear equations of the form \( Ax = b \), where \( A \) is a square sparse matrix that can be either unsymmetric, symmetric positive definite, or general symmetric. MUMPS uses a multifrontal technique which is a direct method based on either the \( LU \) or the \( LDL^T \) factorization of the matrix. MUMPS exploits both parallelism arising from sparsity in the matrix \( A \) and from dense factorizations kernels.

The main features of the MUMPS package include the solution of the transposed system, input of the matrix in assembled format (distributed or centralized) or elemental format, error analysis, iterative refinement, scaling of the original matrix, and return of a Schur complement matrix. MUMPS offers several built-in ordering algorithms, a tight interface to some external ordering packages such as METIS and PORD, and the possibility for the user to input a given ordering. Finally, MUMPS is available in
various arithmetics (real or complex, single or double precision).

The software is written in Fortran 90 although a C interface is available. The parallel version of MUMPS requires MPI for message passing and makes use of the BLAS, BLACS, and ScaLAPACK libraries. The sequential version only relies on BLAS.

MUMPS distributes the work tasks among the processors, but an identified processor (the host) is required to perform most of the analysis phase, to distribute the incoming matrix to the other processors (slaves) in the case where the matrix is centralized, and to collect the solution. The system $Ax = b$ is solved in three main steps:

1. **Analysis.** The host performs an ordering based on the symmetrized pattern $A + A^T$, and carries out symbolic factorization. A mapping of the multifrontal computational graph is then computed, and symbolic information is transferred from the host to the other processors. Using this information, the processors estimate the memory necessary for factorization and solution.

2. **Factorization.** The original matrix is first distributed to processors that will participate in the numerical factorization. The numerical factorization on each frontal matrix is conducted by a main compute processor (determined by the analysis phase) and one or more slave processors (determined dynamically). Each processor allocates an array for contribution blocks and factors; the factors must be kept for the solution phase.

3. **Solution.** The right-hand side $b$ is broadcast from the host to the other processors. These processors compute the solution $x$ using the (distributed) factors computed during Step 2, and the solution is either assembled on the host or kept distributed on the processors.

Each of these phases can be called separately and several instances of MUMPS can be handled simultaneously. MUMPS allows the host processor to participate in computations during the factorization and solve phases, just like any other processor.

For both the symmetric and the unsymmetric algorithms used in the code, a fully asynchronous approach with dynamic scheduling of the computational tasks has been chosen. Asynchronous communication is used to enable overlapping between communication and computation. Dynamic scheduling was initially chosen to accommodate numerical pivoting in the factorization. The other important reason for this choice was that, with dynamic scheduling, the algorithm can adapt itself at execution time to remap work and data to more appropriate processors. In fact, the main features of static and dynamic approaches have been combined and the estimation obtained during the analysis to map some of the main computational tasks has been used; the other tasks are dynamically scheduled at execution time.

The main data structures (the original matrix and the factors) are similarly partially mapped according to the analysis phase.
110.6.1.3 Supernodal Algorithm – SuperLU

The left-looking or column Cholesky algorithm can be implemented for sparse system and can be blocked by using a supernodal formulation. The idea of a supernode is to group together columns with the same nonzero structure, so they can be treated as a dense matrix for storage and computation. Supernodes were originally used for (symmetric) sparse Cholesky factorization (Demmel et al., 1999a). In the factorization $A = LL^T$ (or $A = LDL^T$), a supernode is a range $(r : s)$ of columns of $L$ with the same nonzero structure below the diagonal; that is, $L(r : s; r : s)$ is full lower triangular and every row of $L(r : s; r : s)$ is either full or zero.

Then in left-looking Cholesky algorithm, all the updates from columns of a supernode are summed into a dense vector before the sparse update is performed. This reduces indirect addressing and allows the inner loops to be unrolled. In effect, a sequence of col-col updates is replaced by a supernode-column (sup-col) update. The sup-col update can be implemented using a call to a standard dense Level 2 BLAS matrix-vector multiplication kernel. This idea can be further extended to supernode-supernode (sup-sup) updates, which can be implemented using a Level 3 BLAS dense matrix-matrix kernel. This can reduce memory traffic by an order of magnitude, because a supernode in the cache can participate in multiple column updates (Demmel et al., 1999a). It has been reported in (Ng and Peyton, 1993) that a sparse Cholesky algorithm based on sup-sup updates typically runs 2.5 to 4.5 times as fast as a col-col algorithm. Indeed, supernodes have become a standard tool in sparse Cholesky factorization.

To sum up, supernodes as the source of updates help because of the following (Demmel et al., 1999a):

1. The inner loop (over rows) has no indirect addressing. (Sparse Level 1 BLAS is replaced by dense Level 1 BLAS.)

2. The outer loop (over columns in the supernode) can be unrolled to save memory references. (Level 1 BLAS is replaced by Level 2 BLAS.)

   Supernodes as the destination of updates help because of the following:

3. Elements of the source supernode can be reused in multiple columns of the destination supernode to reduce cache misses. (Level 2 BLAS is replaced by Level 3 BLAS.)

Supernodes in sparse Cholesky can be determined during symbolic factorization, before the numeric factorization begins. However, in sparse LU, the nonzero structure cannot be predicted before numeric factorization, so supernodes must be defined dynamically. Furthermore, since the factors $L$ and $U$ are no longer transposes of each other, the definition of a supernode must be generalized.
Freely available package SuperLU proposed a couple of ways to generalize the symmetric definition of supernodes to unsymmetric factorization (Demmel et al., 1999a). It is now not possible to use Level 3 BLAS efficiently for unsymmetric systems. The implementation in SuperLU performs a dense matrix multiplication of a block of vectors and, although these can not be written as another dense matrix, it has been shown that this Level 2.5 BLAS has most of the performance characteristics of Level 3 BLAS since the repeated use of the same dense matrix allows good use of cache and memory hierarchy.

There are three versions of libraries collectively referred as SuperLU (Demmel et al., 2003),

- **Sequential SuperLU** is designed for sequential processors with one or more layers of memory hierarchy (caches).

- **Multithreaded SuperLU** (SuperLU_MT) is designed for shared memory multiprocessors (SMPs), and can effectively use up to 16 or 32 parallel processors on sufficiently large matrices in order to speed up the computation (Demmel et al., 1999b).

- **Distributed SuperLU** SuperLU_DIST is designed for distributed memory parallel processors, using MPI for interprocess communication. It can effectively use hundreds of parallel processors on sufficiently large matrices (Li and Demmel, 2003).

Parallelizing sparse direct solver for unsymmetric systems is more complicated than parallel sparse Cholesky case. The advantage of sparse Cholesky over the unsymmetric case is that pivots can be chosen in any order from the main diagonal while guaranteeing stability. This lets us perform pivot choice before numerical factorization begins, in order to minimize fill-in, maximize parallelism. precompute the nonzero structure of the Cholesky factor, and optimize the (2D) distributed data structures and communication pattern (Li and Demmel, 2003).

In contrast, for unsymmetric or indefinite systems, distributed memory codes can be much more complicated for at least two reasons. First and foremost, some kind of numerical pivoting is necessary for stability. Classical partial pivoting or the sparse variant of threshold pivoting typically cause the fill-ins and workload to be generated dynamically during factorization. Therefore, we must either design dynamic data structures and algorithms to accommodate these fill-ins, or else use static data structures which can grossly overestimate the true fill-in. The second complication is the need to handle two factored matrices L and U, which are structurally different yet closely related to each other in the filled pattern. Unlike the Cholesky factor whose minimum graph representation is a tree (elimination tree), the minimum graph representations of the L and U factors are directed acyclic graphs (elimination DAGs).

In SuperLU_DIST, a static pivoting approach, called GESP (Gaussian Elimination with Static Pivoting) (Li and Demmel, 1998) is used. In order to parallelize the GESP algorithm, a 2D block-cyclic
mapping of a sparse matrix to the processors is used. An efficient pipelined algorithm is also designed to perform parallel factorization. With GESP, the parallel algorithm and code are much simpler than dynamic pivoting.

The main algorithmic features of SuperLU_DIST solver are summarized as follows (Li and Demmel, 2003):

- supernodal fan-out (right-looking) based on elimination DAGs,
- static pivoting with possible half-precision perturbations on the diagonal,
- use of an iterative algorithm using the LU factors as a preconditioner, in order to guarantee stability,
- static 2D irregular block-cyclic mapping using supernodal structure, and
- loosely synchronous scheduling with pipelining.

In particular, static pivoting can be performed before numerical factorization, allowing us to use all the techniques in good sparse Cholesky codes: choice of a (symmetric) permutation to minimize fill-in and maximize parallelism, precomputation of the fill pattern and optimization of 2D distributed data structures and communication patterns. Users are referred to Li and Demmel (2003) for algorithm details.

### 110.6.2 Performance Study on SFSI Systems

In this section, performance study on popular parallel direct and iterative solvers has been conducted. The purpose is to provide some guidelines on appropriate use of different solvers with the parallel finite simulation framework. Matrix systems from SFSI analysis are used as test cases. The performance investigation uses IA64 Intel-based cluster at SDSC.

#### 110.6.2.1 Equation System
Figure 110.68: Matrices $N = 33081$ (Continuum FEM)
110.6.2.2 Performance Results

Table 110.10: Performance Study on SFSI Systems ($N=33081$)

<table>
<thead>
<tr>
<th>Direct Solvers</th>
<th>Num of CPUs</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUMPS</td>
<td>4</td>
<td>6.0312</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>7.0534</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>5.3472</td>
</tr>
<tr>
<td>SuperLU_DIST</td>
<td>4</td>
<td>20.358</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>13.803</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>13.755</td>
</tr>
<tr>
<td>SPOOLES</td>
<td>4</td>
<td>10.696</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>7.5338</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>6.2448</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iterative Solvers (GMRES)</th>
<th>Num of CPUs</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td>16</td>
<td>96.441</td>
</tr>
<tr>
<td>Parallel ILU(0)</td>
<td>4</td>
<td>277.49</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>276.07</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>135.78</td>
</tr>
</tbody>
</table>

110.6.3 Conclusion

This section presents the parallel solvers implemented in parallel finite element framework. Table 110.10, draws several conclusions about appropriate use of solvers:

- Direct solvers outperform the iterative solver significantly for general cases. It is worthwhile to note that nonsymmetric solvers are used here due to their generality. For special cases such as SPD system, preconditioned CG will show much better performance.

- The Conjugate Gradient method applies only to Symmetric Positive Definite (SPD) system. This puts restriction on the material models we can use in our simulations. Generally speaking, elastic material will yield a SPD stiffness matrix. Plastic material with associative flow rule also satisfies
this category. Plastic material with non-associative flow rule has non-symmetric element stiffness matrix and so will be the global coefficient matrix of the equation system.

- Another category of matrix that deserves attention is the stiffness matrix from softening materials, which possesses at least one negative eigenvalue so the SPD property will be broken. For advanced geo-materials subject to complicated loadings, as the material develops nonlinearity, the condition of stiffness matrix might vary greatly from SPD (elastic phase), to singular (elastic-perfectly-plastic), and non-symmetric non-positive-definite (elastic-non-associative-plastic-softening) cases. This poses another challenge when one tries to use iterative solver for production runs. The unpredictability of stiffness matrix will disable the application of powerful solvers such as Conjugate Gradient for iterative case and Cholesky for direct case.

- The reason why iterative solver exhibits poor efficiency is partly due to the problem size. We can also see from the Table 110.10 that parallel direct solver is not scalable in general. Iterative solver, on the other hand, is more scalable and it is safe to project that when the size of matrix increases, iterative solver has the advantage from the memory requirement point of view.

Parallel equation solving itself is a complicated topic in numerical computing community. In this section, the main purpose is to introduce a robust and generally efficient parallel solver for finite element simulations. So in this sense, parallel direct solvers such as MUMPS and SPOOLES are recommended.
Chapter 111

Solid, Structure – Fluid Interaction

(In collaboration with Dr. Hexiang Wang)
111.1 Chapter Summary and Highlights

111.2 Introduction

The analysis of problems several civil engineering fields often requires a study of fluid-structure systems that are excited by dynamic loads. For example, the evaluation of the structural integrity of nuclear reactor components involves the analysis of structures of complex shape and their interaction with the fluid in which they are embedded Donea et al. (1982). In these cases, both the fluid and the structure might undergo non-linear response.

These needs for safety evaluations have motivated the development of computational methods capable to treat transient, non-linear fluid-structure interaction problems. Donea et al. (1982) presented an arbitrary Lagrangian-Eulerian (ALE) finite element method with automatic and continuous rezoning technique of the fluid mesh. With this method, dynamic response of nuclear reactor under solid fluid interaction has been simulated. Recently Park et al. (2014) examined the modal characteristics of Reactor vessel internals (RVIs) based on scale-similarity analysis with fluid-structure interaction (FSI). It was observed that the added-mass (A-M) model for submerged structures is considerably dependent on mode shapes and natural frequencies. Sigrist et al. (2006) conducted comparative dynamic analysis with FSI modeling for pressure vessel and internals in a nuclear reactor. They proved that the coupling effect is significant, whereas the effect of added-stiffness on global behavior is negligible. Je et al. (2017) stressed that improvement of numerical analysis methods has been required to solve complicated phenomena that occur in nuclear facilities. Particularly, fluid-structure interaction (FSI) behavior should be resolved for accurate design and evaluation of complex reactor vessel internals (RVIs) submerged in coolant. They investigated the FSI effect on dynamic characteristics of RVIs in a typical 1,000 MW e nuclear power plant. Modal analyses of an integrated assembly were conducted by employing the fluid-structure (F-S) model as well as the traditional added-mass model.

Though numerous efforts have been made on simulation of SFI problem in nuclear reactors, to the author’s best knowledge, high fidelity modeling of this dynamic nonlinear phenomena with complex geometry is still unavailable. Full sets of Navier-stokes (N-S) equation is rarely solved for fully solid fluid coupling. Instead, many simplified analysis procedure is adopted: like added-mass (A-M) model (as shown in equation 111.1) Park et al. (2014); Sigrist et al. (2006) and simplified acoustic wave equation Je et al. (2017) (equation 111.2). These simplified methods introduce great modeling uncertainty to the
simulation system.

\[ \{\ddot{u}\} + [C_s]\{\dot{u}\} + [K_s]\{u\} = \{f_e\} + \{ff\} \]
\[ f_f = \int_S \{N_p\}^T \{n\}\{P\}dS = -[M_a]\{\ddot{u}\} \] (111.1)

\[ [M_f]\{\ddot{P}\} + [C_f]\{{\dot{P}}\} + [K_f]\{P\} + \rho [R_{int}]^T \{\ddot{u}\} = 0 \] (111.2)

On the other hand, ALE method, which was originally put forward as a powerful tool for fluid dynamics with deforming boundary Hirt et al. (1974), has been applied to fully solve the coupled N-S equation and solid mechanic equation Le Tallec and Mouro (2001); Murea and Sy (2017). The freedom in moving the fluid mesh offered by the ALE formulation is very attractive. However, it can be overshadowed by the burden of specifying grid velocities, well suited to a particular problem. As a consequence, the practical implementation of the ALE description requires that an automatic mesh displacement prescription algorithm to be supplied. Many methods have been put forward to overcome this difficulty. For example, pseudo-solid method was adopted by Van Loon et al. (2007) and Jasak and Tukovic (2006) came up with a simplified procedure by solving a Laplacian equation 111.3 of grid velocity with finite element discretization.

\[ \nabla \cdot (\gamma \nabla u) = 0 \]
\[ x_{new} = x_{old} + u \Delta t \] (111.3)

Seemingly, introducing these additional equations to specify the movement of fluid mesh can well resolve the inherent problem of ALE method. Based on specified grid velocity, solutions to ALE-formed N-S equations 111.4 can give precise response of fluid flow.

\[ \frac{\partial}{\partial t} (\rho J) = J \nabla \cdot (w - v) \]
\[ \frac{\partial}{\partial t} (\rho v J) = J \nabla \cdot v(w - v) + J(\rho b - \nabla p) \] (111.4)

However, mathematically the system equations (seen in section 111.3.1) to represent physical phenomena of SFI itself is sufficient and complete. Theoretically, no additional equations need to be added to the coupled system. The authors think that the ease gained here by introducing extra mesh movement equations to ALE method is sacrificed with the accuracy of the result of pressure field. Because in equation 111.4, the pressure is dependent on both absolute velocity \( v \) and relative velocity \( (v - w) \).
Different configuration of mesh velocity can result in different relative velocity under the same absolute velocity, which in turn causes different pressure field.

This could be fine for pure fluid dynamics problem, where engineers care more about the fluid flow (i.e. velocity field). However, this may not be good enough for SFI, where hydrodynamic pressure at solid fluid interface is of great of importance for accurate Neumann boundary condition of solid domain. Therefore, precise pressure field is desired and indispensable in high-fidelity simulation of SFI.

The research presented here aims at realistic SFI modeling in nuclear reactors with solutions to fully coupled FSI system (i.e. N-S equations, solid mechanics equations and interface constraint equations). Geometric conformity is also achieved in this work. The great emphasis was put on accurate pressure field at solid fluid interface. A full sets of verification and validation tests are provided to guarantee the reliability of our modeling.

The limitation of current work is that relatively large displacement of solid fluid interface and accompanying Eulerian mesh distortion problem are not well resolved. Further development are needed for these topics.

### 111.3 Theoretical Formulation

#### 111.3.1 Solid Fluid Interaction

The mathematical description of physical phenomenon of solid fluid interaction includes three parts

Van Loon et al. (2007): The response of solid domain $\Omega_s$ is controlled by the theory of general continuum solid mechanics (equation 111.5). The governing equation in fluid domain is Navier-Stokes equation (N-S equations), which basically consists of mass conservation and momentum conservation equation (shown in equation 111.6). In the equations below, symbols $\mathbf{u}$, $\mathbf{\sigma}$, $\mathbf{f}$, $p$, $G$, $\rho$ and $\eta$ denote velocity, Cauchy stress tensor, body force, pressure, solid shear modulus, density and fluid viscosity. $\mathbf{F}$ is deformation gradient tensor defined as $\mathbf{F} = \nabla \chi(X, t)$.

$$\rho_s \frac{d\mathbf{u}}{dt} = \nabla \cdot \mathbf{\sigma} + \rho_s \mathbf{f} \quad \text{in} \ \Omega_s \quad \text{(111.5a)}$$

$$\det(\mathbf{F}) = 1 \quad \text{in} \ \Omega_s \quad \text{(111.5b)}$$

$$\mathbf{\sigma} = G(\mathbf{F} \cdot \mathbf{F}^T - \mathbf{I}) - p_s \mathbf{I} \quad \text{in} \ \Omega_s \quad \text{(111.5c)}$$
\[ \rho^f \frac{\partial \mathbf{u}^f}{\partial t} = \nabla \cdot \mathbf{\sigma}^f + \rho^f f^f \quad \text{in } \Omega_f \quad \text{(111.6a)} \]

\[ \nabla \cdot \mathbf{u}^f = 0 \quad \text{in } \Omega_f \quad \text{(111.6b)} \]

\[ \mathbf{\sigma}^f = 2\eta D(u^f) - p^f I \quad \text{in } \Omega_f \quad \text{(111.6c)} \]

At the solid fluid interface \( D = \partial \Omega_s \cap \partial \Omega_f \), kinematic and dynamic constraints should be met, as shown in equation 111.7.

\[ \mathbf{u}^s - \mathbf{u}^f = 0 \quad \text{in } D \quad \text{(111.7a)} \]

\[ \sigma^s \cdot \mathbf{n} + \sigma^f \cdot \mathbf{n} = 0 \quad \text{in } D \quad \text{(111.7b)} \]

### 111.3.2 Finite Volume Discretization

For general purpose, the standard form of the transport equation for a scalar property \( \phi \) is considered here in 111.8. It is a second order equation as the diffusion term includes the second derivative of \( \phi \) in space. Finite volume discretization will be applied to the integral form regarding to control volume \( V_p \) (equation 111.9) in both spatial and temporal sense Moukalled et al. (2016).

\[ \frac{\partial \rho \phi}{\partial t} + \nabla \cdot (\rho \mathbf{U} \phi) - \nabla \cdot (\rho \Gamma_\phi \nabla \phi) = S_\phi(\phi) \quad \text{(111.8)} \]

\[ \int_{V_p} \frac{\partial \rho \phi}{\partial t} dV + \int_{V_p} \nabla \cdot (\rho \mathbf{U} \phi) dV - \int_{V_p} \nabla \cdot (\rho \Gamma_\phi \nabla \phi) dV = \int_{V_p} S_\phi(\phi) dV \quad \text{(111.9)} \]

- **Spatial discretization**

  The spatial discretization of equation 111.9 includes three parts: discretization of convection term, diffusion term and source term, respectively. Using divergence theorem, the semi-discrete form of convection term at arbitrary control volume \( V_p \) can be given in equation 111.10, where \( F \) denotes the mass flux through the face defined in equation 111.11. In this semi-discrete form, we still need.
to calculate the face value $\phi_f$ and face mass flux $F$ in order to evaluate the whole volume integral of convection term.

$$
\int_{V_p} \nabla \cdot (\rho U \phi) dV = \sum_f S \cdot (\rho U \phi)_f 
$$  \hspace{2cm} (111.10a)

$$
= \sum_f S \cdot (\rho U)_f \phi_f 
$$  \hspace{2cm} (111.10b)

$$
= \sum_f F \phi_f 
$$  \hspace{2cm} (111.10c)

$$
F = S \cdot (\rho U)_f 
$$  \hspace{2cm} (111.11)

The face value $\phi_f$ at face center $f$ can be obtained from face interpolation scheme using $\phi$ value at control volume center $\phi_p$ and value at neighboring volume center $\phi_N$ (equation 111.12).

$$
\phi_f = f_x \phi_p + (1 - f_x) \phi_N 
$$  \hspace{2cm} (111.12)

Here $f_x$ is the interpolation factor defined as the ratio of distances $fN$ and $PN$:

$$
f_x = \frac{fN}{PN} 
$$  \hspace{2cm} (111.13)

Similarly, for diffusion term the semi-discrete form is presented in equation 111.14. With the semi-discrete form, $(\nabla \phi)_f$ still needs to be evaluated to achieve full discretization. For orthogonal mesh, following equation 111.15 can be used to simplify our analysis, where $|d|$ is the magnitude of vector $PN$.

$$
\int_{V_p} \nabla \cdot (\rho \Gamma_\phi \nabla \phi) dV = \sum_f S \cdot (\rho \Gamma_\phi \nabla \phi)_f 
$$  \hspace{2cm} (111.14a)

$$
= \sum_f (\rho \Gamma_\phi)_f S \cdot (\nabla \phi)_f 
$$  \hspace{2cm} (111.14b)
\[ S \cdot (\nabla \phi)_f = |S| \frac{\phi_N - \phi_P}{|d|} \quad (111.15) \]

The source term \( S_\phi(\phi) \) can be a general function of \( \phi \). Before the actual discretization, the source term is first linearized, where \( S_u \) and \( S_p \) also depend on \( \phi \). Then the volume integral of source term can be evaluated with equation 111.17.

\[ S_\phi(\phi) = S_u + S_p \phi \quad (111.16) \]

\[ \int_{V_p} S_\phi(\phi) dV = S_u V_p + S_p V_p \phi_p \quad (111.17) \]

- **Temporal discretization**

Conducting time integration from \( t \) to \( t + \Delta t \) with both sides of equation 111.9 yields equation 111.18.

\[ \int_t^{t+\Delta t} \left[ \frac{\partial}{\partial t} \int_{V_p} \rho \phi dV + \int_{V_p} \nabla \cdot (\rho U \phi) dV - \int_{V_p} \nabla \cdot (\rho \Gamma \phi \nabla \phi) \right] dt = \int_t^{t+\Delta t} (\int_{V_p} S_\phi(\phi) dV) dt \quad (111.18) \]

Substituting the spatial semi-discretization shown above (equations 111.10, 111.14 and 111.17) into equation 111.18 and assuming that the control volumes do not change in time, equation 111.18 can be written as:

\[ \int_t^{t+\Delta t} \left[ \frac{\partial}{\partial t} \int_{V_p} \rho \phi dV + \sum_f F \phi_f - \sum_f (\rho \Gamma \phi) \int_{V_P} S \cdot (\nabla \phi)_f \right] dt = \int_t^{t+\Delta t} (S_u V_p + S_p V_p \phi_p) dt \quad (111.19) \]

Here further temporal discretization are needed to evaluate equation 111.19:

\[ \frac{(\partial \rho \phi)^p}{\partial t} = \frac{\rho^p \phi^p_0 - \rho^0 \phi^0_p}{\Delta t} \quad (111.20a) \]

\[ \int_t^{t+\Delta t} \phi(t) dt = \frac{1}{2}(\phi^0 + \phi^\Delta) \Delta t \quad (111.20b) \]
where
\[ \phi^n = \phi(t + \Delta t) \]
\[ \phi^0 = \phi(t) \]

Assuming that the density and diffusivity do not change in time, the final semi-discrete form including both spatial and temporal discretization can be given in equation 111.21. Since in equation 111.21, \( \phi^n_f, (\nabla \phi)^n_f \) can be expressed with \( \phi \) values at control cell \( \phi_P \) and neighboring cells \( \phi_N \) with equation 111.12 and 111.15. Therefore, equation 111.21 can be finalized into algebraic equation form (equation 111.22).

\[
\frac{\rho_P \phi^n_P - \rho_P \phi^0_P}{\Delta t} V_P + \frac{1}{2} \sum_f F \phi^n_f + \frac{1}{2} \sum_f F \phi^0_f \\
- \frac{1}{2} \sum_f (\rho \Gamma_\phi)_f S \cdot (\nabla \phi)^n_f - \frac{1}{2} \sum_f (\rho \Gamma_\phi)_f S \cdot (\nabla \phi)^0_f = SuV_P + \frac{1}{2} S_p V_P \phi^n_P + \frac{1}{2} S_p V_P \phi^0_P
\]  

(111.21)

\[
a_P \phi^n_P + \sum_N a_N \phi^n_N = R_P
\]

(111.22)

For every control volume, one equation of this form is assembled. The value of \( \phi^n_P \) depends on the values in the neighboring cells, thus creating a system of algebraic equations 111.23, where \([A]\) is a sparse matrix, with coefficients \(a_P\) on the diagonal and \(a_N\) off the diagonal, \([\phi]\) is the vector of \(\phi\)-s for all control volumes and \([R]\) is the source term vector.

\[
[A][\phi] = [R]
\]

(111.23)

111.3.3 Volume of Fluid Method

The physical phenomena of earthquake soil structure interaction in fluid domain is essentially free surface flow. There are three types of problems in the numerical treatment of free boundaries: (1) their discrete representation, (2) their evolution in time, and (3) the manner in which boundary conditions are imposed on them.

Volume of Fluid method is originally put forward by Hirt and Nichols (1981) to solve free surface flow problems (free boundary problems). Different from traditional one-phase method, VOF method considers more general two-phase flow problem. In special case where we take one phase as air and another phase as any liquid that needs to be simulated, then VOF becomes also applicable for free surface flow.
The idea of Volume of Fluid method is to introduce a new field variable - volume fraction value ($\alpha$) for each control volume, which is defined in equation 111.25 where $V_i$ is the total volume of cell $i$ and $V_i^w$ is the volume of water contained in the cell. From the definition, it can be seen that $\alpha$ values range between 0 and 1. If the cell is completely filled with fluid then $\alpha = 1$ and if it is filled with void phase then its value should be 0. In VOF method, momentum equation and continuity equation are solved for one composite fluid phase characterized by volume fraction $\alpha$. The physical properties of this one fluid are calculated as weighted averages based on the volume fractions of two phases in one cell. E.g., the density of any point in the domain is calculated with equation 111.24.

$$ \rho = \alpha \rho_w + (1 - \alpha) \rho_a $$

(111.24)

The evolution of volume fraction $\alpha$ is controlled by an additional convection transport equation 111.26. By solving this equation, the distribution of volume fraction $\alpha$ can be obtained. Then the free fluid interface can be automatically identified as the zone where the volume fraction $\alpha$ ranges between 0 to 1. In this way, the fluid flow can be represented as the redistribution of $\alpha$ along with the time.

$$ \alpha_i = \frac{V_i^w}{V_i} $$

(111.25)

$$ \frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) = 0 $$

(111.26)

The free surface flow solver used here is interFoam implemented based on OpenFOAM. In interFoam, the necessary compression of the phase interface is achieved by introducing an extra artificial compression term into the transport convection equation of $\alpha$, as shown in equation 111.27. $U_r$ is the velocity field suitable to compress the interface. This artificial term is active only in the interface region due to the term $\alpha(1 - \alpha)$.

$$ \frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) + \nabla \cdot (\alpha(1 - \alpha)U_r) = 0 $$

(111.27)

### 111.3.4 Pressure-velocity coupling: PISO algorithm

For incompressible form of the fluid system (equation 111.28), two issues require special attention: non-linearity of the momentum equation and the pressure-velocity coupling Jasak (1996). The non-linear term in equation 111.29 is $\nabla \cdot (UU)$, i.e. velocity is “being transported by itself”. The discretized
form of this expression would be quadratic in velocity and the resulting system of algebraic equations would therefore be non-linear. There are two possible solutions for this problem - either use a solver for non-linear systems, or linearize the convection term.

\[ \nabla \cdot \mathbf{U} = 0 \]  
(111.28)

\[ \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U} \mathbf{U}) - \nabla \cdot (\nu \nabla \mathbf{U}) = -\nabla \mathbf{p} \]  
(111.29)

According to section 111.3.2, convection term can be discretized with equation 111.30, where \( F \), \( a_P \) and \( a_N \) are a function of \( \mathbf{U} \). The important issue is that the fluxes \( F \) should satisfy the continuity equation 111.28. Linearization of the convection term implies than an existing velocity (flux) field that satisfies equation 111.28 will be used to calculate \( a_P \) and \( a_N \).

\[ \nabla \cdot (\mathbf{U} \mathbf{U}) = \sum_f \mathbf{S} \cdot (\mathbf{U})_f (\mathbf{U})_f \]  

\[ = \sum_f F(\mathbf{U})_f \]  

\[ = a_P \mathbf{U}_P + \sum_N a_N \mathbf{U}_N \]  
(111.30)

PISO (Pressure-Implicit Splitting of Operators) procedure proposed by Venier et al. (2017) is used here for pressure-velocity coupling in transient calculations. In order to derive the pressure equation, a semi-discrete form (equation 111.31) of momentum equation is adopted, where pressure gradient term is not discretized at this stage. The \( \mathbf{H}(\mathbf{U}) \) term consists of two parts: the “transport part”, including the matrix coefficients for all neighbors multiplied by corresponding velocities and the “source part” including the source part of the transient term.

\[ a_P \mathbf{U}_P = \mathbf{H}(\mathbf{U}) - \nabla \mathbf{p} \]  

\[ \mathbf{H}(\mathbf{U}) = - \sum_N a_N \mathbf{U}_N + \frac{\mathbf{U}_0}{\Delta t} \]  
(111.31)

In addition, the discrete form of continuity equation is:

\[ \nabla \cdot \mathbf{U} = \sum_f \mathbf{S} \cdot \mathbf{U}_f = 0 \]  
(111.32)

From equation 111.31, \( \mathbf{U} \) can be explicitly expressed with equation 111.33. Then from the explicit solution \( \mathbf{U}_P \), velocities at cell face \( \mathbf{U}_f \) can be calculated through face interpolation (equation 111.34).
Substituting equation 111.34 into equation 111.32 yields the equivalent form of continuity condition (equation 111.35).

\[ U_P = \frac{H(U)}{a_P} - \frac{1}{a_P} \nabla p \]  \hspace{1cm} (111.33)

\[ U_f = \left( \frac{H(U)}{a_P} \right)_f - \left( \frac{1}{a_P} \right)_f (\nabla p)_f \]  \hspace{1cm} (111.34)

\[ \sum_f S \cdot \left[ \left( \frac{1}{a_P} \right)_f (\nabla p)_f \right] = \sum_f S \cdot \left( \frac{H(U)}{a_P} \right)_f \]  \hspace{1cm} (111.35)

The fully discrete form of momentum equation is given in equation 111.36. Equation 111.35 and equation 111.36 constitute the discrete form of incompressible Navier-Stokes system. The face flux \( F \) can be calculated using equation 111.37. When equation 111.35 is satisfied, the face fluxes are guaranteed to be conservative.

\[ a_P U_P = H(U) - \sum_f S(p)_f \]  \hspace{1cm} (111.36)

\[ F = S \cdot U_f = S \cdot \left[ \left( \frac{H(U)}{a_P} \right)_f - \left( \frac{1}{a_P} \right)_f (\nabla p)_f \right] \]  \hspace{1cm} (111.37)

With all the discrete form of system equations prepared, the PISO algorithm can be described as follows:

- The momentum equation is solved first. The exact pressure gradient source term is not known at this stage — the pressure field from the previous time-step is used instead. This stage is called the momentum predictor. The solution of the momentum equation, Eqn. 111.36, gives an approximation of the new velocity field.

- Using the predicted velocities, the \( H(U) \) operator can be assembled and the pressure equation can be formulated. The solution of the pressure equation 111.35 gives the first estimate of the new pressure field. This step is called the pressure correction.

- Eqn. 111.37 provides a set of conservative fluxes consistent with the new pressure field. The velocity field should also be corrected as a consequence of the new pressure distribution. Velocity correction is done in an explicit manner, using Eqn. 111.36. This is the explicit velocity correction stage.
• Eqn. 111.36 reveals that velocity correction consists of two parts: a correction due to the change in the pressure gradient and the transported influence of corrections of neighboring velocities. Explicit velocity correction means that the latter part is neglected. The whole velocity error is assumed to come from the error in pressure term. This is not true. It is therefore necessary to correct the $H(U)$ term and repeat pressure correction and explicit velocity correction stage. In other words, the PISO loop consists of an implicit momentum predictor followed by a series of pressure solutions and explicit velocity corrections. The loop is repeated until a pre-determined tolerance is reached.

111.3.5 Explicit transient algorithm

The algorithm for solid fluid coupling is an explicit segregated approach. The solving of system equation 111.5 in solid domain and system equation 111.6 in fluid domain are performed by RealESSI and InterFoam respectively. A new container called SSFI is implemented in RealESSI to control all the boundary information of solid fluid interface. The algorithm is illustrated in figure 111.1.

Figure 111.1: Flowchart of explicit transient algorithm
Initially all the geometric mapping information of solid fluid interaction is built by SSFI. There are 4 types of geometric mapping in SSFI: Foam node (interface node in fluid domain) maps to foam surface (interface surfaces in fluid domain), ESSI node (interface node in solid domain) maps to foam nodes, foam surface maps to ESSI nodes and foam node maps to ESSI nodes. The specific definitions of these geometric mappings and implementation details can be found in section 111.4.4. These geometric mapping information is indispensable part while conducting the interpolation for interface pressure, velocity and nodal displacement.

Then the equilibrium state of fluid domain is solved first by Interfoam based on the boundary conditions from the response of solid domain at last time step. Here the PIMPLE algorithm Chen et al. (2014) is implemented in InterFoam to couple pressure and velocity, which is a hybrid of the PISO and SIMPLE (Semi-Implicit Method for Pressure-Link Equations) algorithms. In the PIMPLE loop, the transport equation of volume fraction (equation 111.26) is firstly calculated based on existing velocities and surface fluxes. Following that, there is an implicit momentum predictor and several pressure-velocity correctors. After the fluid domain achieves equilibrium for the new time step, the pressures at fluid interface are transformed to equivalent nodal force and applied at the solid interface. This operation is called interface pressure updating.

With updated nodal force at solid interface and some other transient boundary conditions, the response of solid domain for the new step is obtained through transient analysis of RealESSI (Jeremić et al. (1988-2021)). Based on the latest location of solid interface, the geometry of fluid interface is updated correspondingly to make sure the geometric conformity of both domains.

No slip boundary condition is adopted here for the velocity at fluid interface. Therefore, the new boundary velocities are also calculated from the response of solid domain. In addition, it is crucially important to update volume fractions so that mass conservation is guaranteed. The detailed information about interpolation and updating of these physical fields (i.e. pressure, velocity and volume fractions) is presented in section 111.4.6 and 111.4.5.

For the above explicit transient algorithm, both solid domain and fluid domain can individually reach equilibrium states through iterations at both sides with updated interface boundary conditions. However, this two equilibrium states are not achieved at the same time. Current equilibrium state of solid domain matches with the equilibrium state of fluid domain at the last time step. Therefore, the algorithm is only valid and accurate when time step is small enough. Different time step lengths in solid domain and fluid domain can also be handled through Shepherd method. SSFI can go through transient analysis in solid domain and fluid domain alternatively according to different time step length.
111.4 Implementation Details

111.4.1 Installation of OpenFoam

The official website of Openfoam: www.openfoam.com/download/install-source.php gives detailed instruction about how to build Openfoam from source on different operating systems. Installation of customized version of OpenFOAM, that can interact, connect to Real-ESSI is described in some detail in chapter 209.7 on page 1304.

Note that after installation of Openfoam, you need to source the OpenFOAM environment by executing, e.g. for bash shells and OpenFoam version v1706:

```
source /OpenFOAM/OpenFOAM-v1706/etc/bashrc
```

111.4.2 Integrated Preprocessor-gmFoam

**gmFoam** is designed as an integrated preprocessor developed for analysis of solid fluid interaction. It enables user to build an integrated geometric model (including both solid part and fluid part) in Gmsh and generate input files for both RealESSI and interFoam in a very easy way. Some simple examples can be seen in figure 111.2. Its main functions are listed below:

![Figure 111.2: Numerical models built with gmFoam](image-url)
• **Mesh separation**

Note that FVM mesh supported by Openfoam is very flexible. It can be any types of convex polyhedron as shown in figure 111.3). And the geometric description of FVM mesh required by Openfoam is totally different from that of FEM mesh. The description of FVM mesh is face-based. Faces (including both boundary surfaces and exterior surfaces) of control volumes are defined by a list a point IDs that consist of the face. Also the owner cell ID and neighbor cell ID of each face have to be specified. In contrast, the description of FEM mesh is element-based. After defining all the nodes in the model, the element is described by a list of node IDs. gmFoam supports both types of mesh description. User can build an integrated geometric model in Gmsh and define solid part and fluid part as different physical volume groups. After meshing it, gmFoam can separate the mesh information about solid domain and fluid domain and transfer them to FEM mesh and FVM mesh, respectively.

![Mesh for FVM](image)

**Figure 111.3:** Mesh for FVM

• **GmESSI incorporation**

gmFoam perfectly incorporates current RealESSI preprocessor GmESSI so that it can quickly generate ESSI input files for the simulation of solid part. Also it has the capability to quickly generate input files for Openfoam. Currently the input file organization of Openfoam is very complicated. Several folders and files are needed to prepare in order to complete a very simple simulation. But with gmFoam all these basic information can be written in one single file with suffix as `.gmfoam` and gmFoam will automatically parse the content inside and produce all the input folders and files. In addition, with the help of physical group, gmFoam enables user to set different boundary conditions in a very convenient way. This is extremely helpful when we conduct solid fluid interaction
analysis for big models with complicated boundary conditions.

- Interface geometry extraction

  gmFoam can extract the geometric information of solid-fluid interface and write down corresponding information as input files. The information is used to initialize SSFI object and build important geometric interface mapping.

- Support discontinuous mesh

Discontinuous mesh is supported. This is very important, especially for large-scale simulation for solid fluid interaction. Usually refined mesh is needed for fluid part to get accurate enough result using VOF. Discontinuous mesh enables us to arbitrarily refine fluid mesh without changing solid mesh. Therefore, increase of computational efforts in solid part can be avoided. Figure 111.4 shows a model with discontinuous mesh (fluid mesh size : solid mesh size=1:3).

![Numerical model with discontinuous mesh](image)

**Figure 111.4: Numerical model with discontinuous mesh**

### 111.4.3 Interface Domain-SSFI

An interface class SSFI was implemented in RealESSI to couple computations between solid domain and fluid domain. SSFI behaves like a container (called interface domain) to control geometric mapping, sequence of computation and boundary data interpolation and transmission. SSFI class contains and
operates the objects of four base classes: ESSINode, ESSISurface, FoamNode, FoamSurface. Two core member functions: SSFI::FoamToESSIUpdate(double t) and SSFI::ESSIToFoamUpdate(Domain* theDomain, double t) are designed to perform all necessary updates on solid domain and fluid domain, respectively.

111.4.4 Geometric Mapping

Following geometrical mappings are built and maintained in SSFI.

- **Foam node mapping to Foam surfaces**
  For each foam node in solid fluid interface, this mapping returns all the IDs of its surrounding foam surfaces. For example, as shown in figure 111.5(a), foam node 5 is mapped to foam surface 1,2,3 and 4.

  \[\text{Foam node} \rightarrow \text{Foam Surfaces} \]

  \[
  5 \quad (1,2,3,4)
  \]

- **ESSI node mapping to Foam nodes**
  For each ESSI node in solid fluid interface, this mapping returns all the IDs of its surrounding foam nodes within certain search radius (by default the radius is set as 0.1 meters). If the mesh size is very refined, reducing the searching radius helps to improve accuracy.

- **Foam face mapping to ESSI nodes**
  For each foam face in solid fluid interface, this mapping returns 4 vertex node IDs of an ESSI surface that contains the center of this foam face. Like in figure 111.5(b), foam face 1 (consists of foam node 1,2,3 and 4) is mapped to ESSI nodes 1,2,3 and 4.

  \[\text{Foam face} \rightarrow \text{ESSI nodes} \]

  \[
  1 \quad (1,2,3,4)
  \]

- **Foam node mapping to ESSI nodes**
  For each foam node in solid fluid interface, this mapping returns 4 vertex node IDs of an ESSI surface that contains the Foam node. In figure 111.5(b), foam node 1 is mapped to ESSI nodes 1,3,8 and 7.

  \[\text{Foam node} \rightarrow \text{ESSI nodes} \]

  \[
  1 \quad (1,3,8,7)
  \]
111.4.5 SFI Interpolation

After building all of these geometric mapping, interpolation scheme is also needed to fully determine the values of interface variables (pressure, velocity and displacement) and update these values during the interaction process. There are three types of interpolation and updating involved here:

- **Pressure interpolation**
  The interpolation and updating of interface pressure happen during the process of FOAM to ESSI updating. After the fluid domain achieves its equilibrium, the new pressure values at interface foam faces need to be interpolated and transferred to corresponding ESSI nodes. The pressure interpolation scheme is following:

  Firstly, using the mapping from foam node to foam surfaces, pressure at each interface foam node is calculated by taking the average pressure values of its surrounding foam surfaces. Like in figure 111.5(a), the pressure at interface foam node 5 is the average value of pressure at interface foam surface 1,2,3 and 4. Then according to the mapping from ESSI node to foam nodes, the updated pressure at interface ESSI node is calculated as the average pressure value of corresponding foam nodes.

- **Velocity interpolation**
  Velocity interpolation takes place during the process of ESSI to FOAM updating. After RealESSI

![Figure 111.5: Geometric mapping in SSFI](image)
concludes transient analysis for solid domain, the velocities at interface ESSI nodes need to be fed back to corresponding interface foam surface. The velocity interpolation scheme is following:

The mapping from Foam face to ESSI nodes can give four vertex interface ESSI node ID. For example, in figure 111.5(b), foam face 1 is mapped to four ESSI nodes (1,2,3,4). With coordinates of foam face center and four ESSI nodes already known, inverse isoparametric mapping Hua (1990) is used here to determine the local coordinates (ζ, η) of foam face center. The formula of inverse isoparametric mapping is shown below:

![Figure 111.6: Illustration of isoparametric mapping Hua (1990)](image)

\[
\begin{bmatrix}
    a_1 & a_2 \\
    b_1 & b_2 \\
    c_1 & c_2 
\end{bmatrix} = \begin{bmatrix}
    1 & -1 & 1 & -1 \\
    -1 & 1 & 1 & -1 \\
    -1 & -1 & 1 & 1 
\end{bmatrix} \begin{bmatrix}
    x_1 & y_1 \\
    x_2 & y_2 \\
    x_3 & y_3 \\
    x_4 & y_4 
\end{bmatrix} = d_1 \begin{bmatrix}
    x_1 & y_1 \\
    x_2 & y_2 \\
    x_3 & y_3 \\
    x_4 & y_4 
\end{bmatrix}
\]

(111.38)

\[
d_1 = 4x - (x_1 + x_2 + x_3 + x_4)
\]

\[
d_2 = 4y - (y_1 + y_2 + y_3 + y_4)
\]

(111.39)

A compact notation to represent the determinant of a 2 matrix is introduced as

\[
r_s = \begin{vmatrix}
    r_1 & s_1 \\
    r_2 & s_2 
\end{vmatrix} = r_1s_2 - r_2s_1
\]

(111.40)
where \( r, s = a, b, c, d \). Notice that \( rs = -sr \).

With all the notations well defined, the solutions to local coordinate given by Hua (1990) are shown below:

- \( a_1a_2a_3a_4 \neq 0 \)
  \[
  \begin{cases}
    a_b \zeta^2 + (c_b + d_a) \zeta + d_c = 0 \\
    \eta = (a_d + b_a \zeta)/a_c \\
  \end{cases}
  \]  
  where \( \zeta \in [-1, 1] \)

- \( a_1 = 0 \) and \( a_2a_3 \neq 0 \)
  \[
  \begin{cases}
    a_b \zeta^2 + (c_b + d_a) \zeta + d_c = 0 \\
    \eta = (a_d + b_a \zeta)/a_c \\
  \end{cases}
  \]  
  where \( \zeta \in [-1, 1] \)

- \( a_2 = 0 \) and \( a_1b_2 \neq 0 \)
  \[
  \begin{cases}
    a_b \zeta^2 + (c_b + d_a) \zeta + d_c = 0 \\
    \eta = (a_d + b_a \zeta)/a_c \\
  \end{cases}
  \]  
  where \( \zeta \in [-1, 1] \)

- \( a_1a_2 \neq 0 \) and \( a_b = 0 \)
  \[
  \begin{cases}
    \zeta = (a_1 d_c)/(b_1 a_c + a_1 a_d) \\
    \eta = a_d/a_c \\
  \end{cases}
  \]

- \( a_1a_2 \neq 0 \) and \( a_c = 0 \)
  \[
  \begin{cases}
    \zeta = a_d/a_b \\
    \eta = (a_1 d_b)/(c_1 a_b + a_1 a_d) \\
  \end{cases}
  \]

- All other conditions
  \[
  \begin{cases}
    \zeta = d_c/(a_1 d_2 + b_c) \\
    \eta = b_d/(a_2 d_1 + b_c) \\
  \end{cases}
  \]

After obtaining local coordinates \((\zeta, \eta)\) corresponding to foam surface center, isoparametric mapping can be conducted through the shape function of 4-node quadrilateral element. The interpo-
lated velocity can be calculated with equation 111.41.

\[
N_1 = \frac{1}{4}(1 - \zeta)(1 - \eta) \\
N_2 = \frac{1}{4}(1 + \zeta)(1 - \eta) \\
N_3 = \frac{1}{4}(1 + \zeta)(1 + \eta) \\
N_4 = \frac{1}{4}(1 - \zeta)(1 + \eta)
\]

\[v = N_1 v_1 + N_2 v_2 + N_3 v_3 + N_4 v_4\]  \hspace{1cm} (111.41)

- Displacement interpolation

Displacement interpolation also takes place during ESSI to FOAM updating. In order to meet the geometric conformity, the Eulerian mesh of fluid domain should dynamically move along with the real-time response of solid domain. Therefore, it is necessary to interpolate the displacement of interface ESSI nodes to interface foam nodes. The displacement interpolation scheme is similar to velocity interpolation scheme and shown below:

The mapping from Foam node to ESSI nodes can give four vertex interface ESSI node ID. Inverse isoparametric mapping is first performed to compute local coordinate of Foam node. Then displacement of foam node is interpolated from displacement of four vertex ESSI nodes through isoparametric mapping.

### 111.4.6 Mass Conservation

There are two levels of mass conservation conditions needed to be satisfied during the SFI.

One is local level: Regarding each control volume, the amount of fluid flows in should equal to the amount of fluid that flows out for incompressible fluid. This local mass conservation condition is mathematically represented by continuity equation in equation 111.6 and can be approximately met through finite volume discretization. Another level of mass conservation is global level: For a closed fluid domain (no fluid transfer with other external fluid system), the total amount of fluid should keep constant during SFI. The global level of mass conservation is trivial for pure fluid flow system with static Eulerian boundary mesh. Since for this kind of flow system, global level of mass conservation is free and automatically holds based on local level of mass conservation.

However, for flow system with deforming boundary, especially when Lagrangian movement of Eulerian mesh is involved (like Arbitrary Lagrangian method Souli and Zolesio (2001)), the local level of mass
conservation does not guarantee the mass conservation of global level. Demirdžić and Perić (1988) pointed out that for moving mesh, one more conservation equation so called space conservation needed to be solved simultaneously with the mass, momentum and energy conservation equations. Otherwise artificial mass addition or reduction is generated which may cause the solution to be greatly in error.

The space conservation law is expressed by equation 111.42, where $J$ is the determinant of the metric tensor and $v_g$ is the grid velocity of the mesh. Correspondingly, the finite volume discrete evaluation of volume integral of equation 111.42 can be given in equation 111.43, where $\delta V = V^n - V^0$ is the change of cell volume during $\Delta t$, $v_g(f)$ is the mesh velocity of cell face and $S$ is cell face vector.

$$\frac{\partial J}{\partial t} - J \nabla \cdot v_g = 0$$  \hspace{1cm} (111.42)

$$\frac{V^n - V^0}{\Delta t} = \sum_f v_g(f) \cdot S$$  \hspace{1cm} (111.43)

In our implementation, consider the coupling with VOF method and our SFI simulation is generally for small deformation of fluid boundary, a simplified procedure is adopted here to guarantee the global level of mass conservation: Only the location of foam nodes at solid fluid interface is updated with Lagrangian motion of corresponding ESSI nodes while interior foam nodes remain static. After geometry updating of the mesh, the volume of foam cells at interface is re-evaluated and volume fraction values are also updated according to equation 111.44 to ensure the mass conservation. Then the transport equation 111.27 is solved based on new $\alpha$ values.

$$\alpha = \int \frac{\alpha^0 dV^0}{dV} = \frac{\alpha^0 V^0}{V}$$  \hspace{1cm} (111.44)
Part 200

Software and Hardware Platform: Design, Development, Procurement and Use
Chapter 201

The Real ESSI Simulator System
201.1 Chapter Summary and Highlights

201.2 Introduction to the Real-ESSI Simulator System

The Real-ESSI Simulator (Realistic Modeling and Simulation of Earthquakes, and/or Soils, and/or Structures and their Interaction) is a software, hardware and documentation system for high performance, sequential or parallel, time domain, linear or nonlinear, elastic and inelastic, deterministic or probabilistic, finite element modeling and simulation of

- statics and dynamics of soil,
- statics and dynamics of rock,
- statics and dynamics of structures,
- statics of and dynamics of soil-structure systems,
- dynamics of earthquakes, and
- dynamic earthquake-soil-structure interaction.

The Real-ESSI Simulator systems is used for design and for assessment of static and dynamic behavior of infrastructure objects, including buildings, bridges, dams, nuclear installations, tunnels, etc.

**Design**: Multiple linear elastic load cases can be combined and design quantities, sectional forces exported for design.

**Assessment**: Practical, realistic, inelastic, nonlinear load staged analysis, with accurate modeling of elastic and inelastic, nonlinear components, and with all the simulation, algorithmic features available, as listed below, is performed to assess design, current safety margins and economy of objects.

The Real-ESSI Simulator is developed at the University of California, Davis, in collaboration and with partial financial support from the USDOE, USNRC, USNSF, USBR, USFEMA, CalTrans, CNSC-CCSN, UN-IAEA, Shimizu, Private Donors, etc. The Real-ESSI Simulator develops methods and models that inform and predict rather than (force) fit.

The Real-ESSI Simulator systems consists of the Real-ESSI Program, Real-ESSI Pre-Processing and Post-Processing tools, Real-ESSI Computer and Real-ESSI Notes.

201.2.1 Real-ESSI Program

The Real-ESSI program is a general purposes finite element program that features models and methods for analyzing static and dynamic behavior of Civil Engineering objects, such as buildings, bridges, dams, nuclear installations, tunnels, etc.
201.2.2 Real-ESSI Pre-Processing tools

The Real-ESSI Pre-Processing tools are a set of programs, scripts and modules that are used to develop Real-ESSI models. Mesh generation relies on Gmsh (Geuzaine and Remacle, 2009) and our own plugins Gm-ESSI, while there are also mesh translators from other input formats to Real-ESSI input format/language.

201.2.3 Real-ESSI Post-Processing tools

The Real-ESSI Post-Processing tools rely on Paraview (Ayachit, 2015) visualization platform, with our own plugins, as well as on a number of programs and scripts to visualize output using matlab, python, etc.

201.2.4 Real-ESSI Computer

The Real-ESSI Computer can be any single CPU, multiple CPU, and/or cluster of single/multiple CPUs computers using Linux operating system. The reason for using Linux is that state of the art development tools are available, and that Linux is used on virtually all large supercomputers, so that the very same sources can be compiled and executables developed for small desktop computers, large server computers, local clusters of computers and large supercomputers. There is a possibility to build Real-ESSI and create executables on Macintosh and Windows platforms, as long as standard software tools and compiles, C++ and Fortran, are available. However, such work is not currently pursued.

201.2.5 Real-ESSI Notes


201.2.6 Real-ESSI Name

The Real-ESSI Simulator System name is based on an acronym: Realistic Modeling and Simulation of Earthquakes, and/or Soils, and/or Structures and their Interaction. Pronunciation of Real-ESSI is similar to "real easy", as in "as easy as pie". Translation of Real ESSI to other languages (languages of developers and users) is also available: Vrlo prosto, Просто к’о пасуљ, Muy fácil, Molto facile, Ελληνικά εύκολο, बहुत ही आसान, أسان واقعی, Très facile, Вистински лесно, Wirklich einfach, سهل جداً.
Chapter 202

Object Oriented Software Platform Design

(In collaboration with Dr. Guanzhou Jie)
202.1 Chapter Summary and Highlights

202.2 Object-Oriented Design Basics

Booch (1994); Gamma et al. (1995); Coplien (1992); Koenig (1989 - 1993); Stroustrup (1986); Stroustrup (1994); Ellis and Stroustrup (1990); Johnson (1994); Felippa (1992a); Dubois-Pelerin (1992); Dubois-Pelierin and Zimmermann (1993); Raphael and Krishnamoorthy (1993); Menéntrey and Zimmermann (1993); Zimmermann et al. (1992a); Donescu and Laursen (1996); Zimmermann and Eyheramendy (1996); Eyheramendy and Zimmermann (1996); Jeremić and Sture (1998); Forde et al. (1990); Miller (1991); Scholz (1992); Fenves (1990); Eyheramendy (1997); Dubois-Pelierin and Zimmermann (1992); Zimmermann et al. (1992b); Dubois-Pelierin and Pegan (1998); Eyheramendy and Zimmermann (2001);

Veldhuizen (1995a); Veldhuizen (1995b); Veldhuizen (1996); Veldhuizen and Jernigan (1997);
Archer (1996); Archer et al. (1999); McKenna (1997);

202.3 Object-Oriented Design of the Plastic Domain Decomposition (PDD)

202.3.1 Introduction

This section describes the object oriented design of the proposed PDD algorithm and its implementation into MOSS library framework. At the beginning of this section, the Object-Oriented approach to programming the Finite Element Method is reviewed based on the existing (as of 2005) implementation of OpenSees. Object-Oriented parallel design is then extended from the existing framework. Parallel algorithm adopts Main-Follower paradigm and the new design of data structures have strictly followed the Object-Oriented principle using C++ language. External utility libraries such as ParMETIS and PETSc have been incorporated to provide seamless parallel numerical manipulations including partitioning/repartitioning and equation solving.

In this chapter, the algorithm overview will be presented first. Then the implementation details in C++ will follow. The challenges of achieving load balancing in parallel Finite Element simulation have been divided into two parts, global level equation solving and constitutive level iterations. This research presents the PDD algorithm to demonstrate how to balance each stage systematically in applications.
202.3.2 Object-Oriented Parallel Finite Element Algorithm

Parts of OpenSees software framework (McKenna, 1997) have been used in this chapter. Object-Oriented design of OpenSees enables software reuse that greatly shortens the development life cycle of application codes.

OpenSees is comprised of a set of classes and objects that represent models perform computations for solving the governing equations, and provide access to processing results. There are four types of class objects in OpenSees McKenna (1997).

- **Modeling Classes** are used to create the Finite Element Model Classes for a given problem.

- **Finite Element Model Classes** are used to describe the finite element model and to store the results of the analysis performed on the model. Main class abstractions used in OpenSees are **Node, Element, Constraint, Load** and **Domain**. The relationship amongst these classes can be shown using the class diagram Figure 202.2 using the Rumbaugh notation as shown in Figure 202.1 Rumbaugh et al. (1991).

- **Analysis Classes** are used to perform the finite element analysis, i.e., to form and solve the global system of equations

- **Numerical Classes** are used to handle numerical operations in the solution procedure. Also included in this category are data structure classes such as **Vector, Matrix and Tensor**.

**Figure 202.1:** Rumbaugh Notation of-Object Oriented Design
Figure 202.2: Class Diagram of Finite Element Model Classes
202.3.2.1 Modeling Classes

The modeling classes are responsible of creating the necessary components of the finite element model, such as nodes, elements, loads and constraints. There are a number of approaches proposed by various researchers. In some works, the user has the responsibility to create the finite element model in a single driver-type file Ross et al. (1992); Zeglinski et al. (1994); Cardona et al. (1994). In other works, an input file containing the model data is used to be read by the main program to create the model Forde et al. (1990); Dubois-Pelerin et al. (1992); Dubois-Pelerin and Zimmermann (1993); Menentrey and Zimmermann (1993). Graphical interface for building models visually has also been proposed Ostermann et al. (1995); Mackie (1995).

In this research, the existing ModelBuilder interface class is reused to facilitate the finite element model construction. As shown in Figure 202.2, the ModelBuilder is associated with a single finite element Domain object. The interface (pure virtual function) buildFE_Model() must be redefined depending on the specific type of finite element model users want to build.

In parallel processing, PartitionedModelBuilder is used instead, in which the building process includes higher level control of building Subdomains from the PartitionedDomain. For each Subdomain, the PlaneFrameModelBuilder-type is invoked to build the finite element model on each Subdomain.

The Object-Oriented interface design of ModelBuilder through pure virtual function provides a consistent framework from which all kinds of engineering model can be readily extended.

202.3.2.2 Finite Element Model Class

In most of the works presented, main class abstractions used to describe a finite element model are: Node, Element, Constraint, Load and Domain Forde et al. (1990); Zimmermann et al. (1992a); Dubois-Pelerin et al. (1992); Dubois-Pelerin and Zimmermann (1993); Menentrey and Zimmermann (1993); Pidaparti and Hudli (1993); Cardona et al. (1994); Chudoba and Bittnar (1995); Zahlten et al. (1995); Rucki and Miller (1996).

Node The most important feature of the Node class is the associativity with DOF class which contains the degree of freedoms of any specific instance of the Node class. The response quantities such as displacements of each DOF object will be stored in the Node class. Routines are available to set/get those solution quantities.

Element The functionality of an Element object is to provide the tangent stiffness, mass and the residual force corresponding to current loadings. Element class contains reference to its associated Node objects.
Element class is one of the most fundamental abstractions in Object-Oriented finite element software design. In this research, Element also acts as a container for material models, which is critical for simulations with nonlinear materials. Chudoba and Bittnar (1995) proposed a MaterialPoint object which is associated with GaussPoint object. In Zahlten et al. (1995), class abstractions such as cross section, material point, material law, yield surface, hardening rule and flow rule are introduced to model complicated materials within the Element class in an Object-Oriented flavor.

Jeremić and Yang (2002) present the complete formulation of Template3Dep material class, which is wrapped inside the Element class to enable a consistent interface for complex elastic-plastic material modeling.

Constraint  There are two types of constraints in finite element simulations,

1. Single-Point constraints, which are applied to a specific DOF object;

2. Multi-Point constraints, which describe the relationship between more than one DOF objects.

In current implementation of OpenSees (version from 2005), the two classes SP_Constraint and MP_Constraint are designed but they do not handle the constraints. These two classes are responsible of setting up relations between Nodes and the constrained DOF_Groups. This will be covered shortly in Analysis class design.

Load  There are also two types of loads that are commonly seen in finite element analysis:

1. node loads that act on specific Nodes;

2. element loads that act on specific Elements, which can be due to body forces, surface tractions, initial stresses and temperature gradients.

In the current (version version 2005) implementation of OpenSees, three extra classes are introduced to handle loading conditions, LoadPattern, NodalLoad and ElementLoad. The LoadPattern is a container class that provides methods in its interface to allow NodalLoad and ElementalLoad objects to be created, traversed and removed. As shown in Figure 202.2, each NodalLoad or ElementalLoad object is associated with a Node or Element object and is responsible of applying nodal or elemental loads to that object.

Domain  The Domain class is the most important container class that is responsible of holding all components of the finite element model, i.e. all the Nodes, Elements, Constraints and Loads. Domain
class acts as the interface between Analysis class and all the individual components of the finite element model. The interface of Domain enables component creation, information access and component removal.

202.3.2.3 Analysis

The Analysis class (McKenna, 1997) is responsible for forming and solving the governing equations for the finite element model. As for nonlinear problems, incremental solution techniques are required and iterative schemes such as Newton-Raphson needed to solve the nonlinear system of equations.

For incremental solution algorithm, the computational tasks are more involved for the finite element analysis.

- Assign equation numbers and map these to the nodal DOFs. This step can be of significant influence on the bandwidth of the coefficient matrix, which is inherently sparse due to the compact support of finite element formulation.
- Form the matrix equations using contributions from elements and nodes.
- Apply the constraints, which may involve transforming the element and nodal contributions or adding additional terms and unknowns to the matrix equations depending the method employed to handle constraints.
- Solve the matrix equations for the incremental nodal displacements.
- Determine the internal state and stresses in the elements.

The Object-Oriented design of the Analysis class is done by firstly breaking down the main tasks performed in a finite element analysis, abstracting them into separate classes, and then specifying the interface for these classes. The Analysis class is an aggregation of all the sub-functionality classes of following types:

1. SolutionAlgorithm class describes the complete computation procedure (steps) in the analysis.

2. AnalysisModel is a container class that stores and provides access to the following types of classes:
   
   (a) DOF_Group class represents the DOF at the Nodes or new DOF introduced into the analysis to enforce the constraints;

   (b) FE_Element class represents the real Elements in the Domain or they are introduced to add stiffness and/or load to the system of equations in order to enforce the constraints.

   It is worthwhile to mention that the FE_Elements and DOF_Groups have very important
design implications although they might seem redundant at the first sight. The significance comes from the facts that

i. they record the mapping between DOFs and equation numbers in the global system which greatly simplifies the interfaces of Node and Element class;

ii. they also provide the interfaces for forming tangent and residual vectors which are used to form the global system of equations;

iii. they are major utility classes of handling constraints.

3. Integrator defines how the FE_Elements and DOF_Groups contribute to the system of equations and how the response quantities should be updated given the solution to the global system of equations.

4. ConstraintHandler handles the constraint by creating adequate FE_Elements and DOF_Groups.

5. DOF_Numberer maps the equation number to the DOFs in the DOF_Groups.


The aggregation of the Analysis object is shown in Figure 202.3.
Figure 202.3: Class Diagram of Analysis Aggregation
Traditional program flow diagrams are used to describe how the nonlinear finite element algorithm control flow implemented in OpenSees. These flow charts are organized as following:

Figure 202.4 shows the overall analysis algorithm flow for nonlinear finite elements. Then this overall analysis flow is broken down into detailed subroutines, such as `theIntegrator::newStep()` and `theAlgorithm::solveCurrentStep()`.

- Figure 202.5 explains in detail the function flow of `theIntegrator::newStep()`, which illustrates the (fairly standard) incremental finite element solution techniques implemented in OpenSees.

- Figure 202.6 shows the function flow of forming the tangent stiffness matrix, which is a loop assembling the global equation system involved in function `theIntegrator::newStep()`.

- Figure 202.7 further describes the Newton-Raphson type iterative solution schemes involved in function `theAlgorithm::solveCurrentStep()`.
Figure 202.4: Overall Algorithm Flow Chart for Nonlinear Finite Element Analysis
Figure 202.5: Detailed View: theIntegrator::newStep() - Incremental Solution Techniques for Nonlinear Finite Element Analysis
Figure 202.6: Detailed View: Assembly of Global Equation System in `theIntegrator::newStep()`
Figure 202.7: Detailed View: `theAlgorithm::solveCurrentStep()` - Newton-Raphson Iterative Schemes for Nonlinear Finite Element Analysis
Finite element simulations inherently are element-based operations, so little modification is needed to parallelize the algorithms described above, although special attention has to be paid to synchronize the computation among different processors. Figure 202.8 shows the activity flow for parallel nonlinear finite element simulations.

![Parallel Activity Flow Diagram of Nonlinear Finite Element Analysis](image-url)
202.3.2.4 Object-Oriented Domain Decomposition

There are three most notable designs of Domain Decomposition method in literature McKenna (1997).

1. **Sause and Song (1994)** presents an Object-Oriented design for linear static analysis using sub-structuring. The interface is restricted to substructuring or FETI Farhat and Roux (1991b) only, and repeated geometry limits the applicability of this design to large problems.

2. **Archer (1996)** proposes a SuperElement class that is a subclass of Element and has a Domain class aggregated. This design is conceptually inappropriate and it results excessive method calls as methods that are for the SuperElement must be called by the SuperElement on the associated Domain McKenna (1997).

3. **Miller and Rucki (1993)** introduces the Partition class which is associated with an Algorithm class. The Algorithm class is responsible for updating the state of a Partition so that it will be in equilibrium. Again, this design is good for substructuring type Domain Decomposition analysis. If we want to solve a problem before the interface solution can be determined, the design fails.

The current design of OpenSees McKenna (1997) proposes many new classes to facilitate flexible Object-Oriented Domain Decomposition. The main abstractions include PartitionedDomain, DomainDecompAnalysis, DomainDecompSolver Subdomain, DomainPartitioner and GraphPartitioner. The class diagram is shown in Figure 202.9.
Figure 202.9: Class Diagram of Domain Decomposition Analysis
PartitionedDomain  The PartitionedDomain class is a subclass of Domain whose objects can be partitioned into Subdomain objects. Aside from common functionality inherited from Domain, PartitionedDomain class provides methods for partitioning the Domain and retrieving information from Subdomains. PartitionedDomain the aggregation of Subdomains and is the major containing class in main compute process.

DomainPartitioner  The DomainPartitioner class is responsible for performing the actual operation to split the PartitionedDomain. The DomainPartitioner will call its associated GraphPartitioner to partition the PartitionedDomain. It also provides the methods to migrate Elements, Nodes, Constraints, Loads amongst Subdomains.

DomainPartitioner is one of the most important utility class in OpenSees in the sense that all partitioning routine and data migration operations will be rooted from this class.

GraphPartitioner  This class utilizes external graph partitioner to color the finite element connectivity graph, which will be constructed from the PartitionedDomain. The result will be fed back to DomainPartitioner to facilitate subsequent data distribution.

GraphPartitioner introduces graph partitioning into OpenSees and the main functionality of this class is to call API and provide necessary data structures from the specific application.

Subdomain  The Subdomain class inherits from both Element and Domain. This has a dual-level design implication:

1. for the top PartitionedDomain, superclass Element is a proxy class of subclass Subdomain, in the sense that all the relevant operations on Elements invoked by PartitionedDomain will be redirected to the specific Subdomain;

2. for any specific Subdomain, it inherits all the interfaces of Domain to do all the computations required by PartitionedDomain.

202.3.2.5 Parallel Object-Oriented Finite Element Design

There has been much effort by researchers on parallel implementation of finite element computations, which can be categorized into either domain decomposition methods or parallel equation solving.

Domain decomposition is favored by many researchers due to its nice “divide and conquer” approach. The subdomains in the domain decomposition method are each assigned to a processing node, which will perform all the computations on that subdomain.
Of the domain decomposition methods, the substructuring method has been the most popular choice although other methods such as iterative substructuring Carter et al. (1989) and FETI (Finite Element Tearing and Interconnecting) Farhat and Roux (1991b); Farhat and Crivelli (1994) have also been used. In the substructuring method presented, static condensation is typically performed on the assembled system of equations.

Earlier works on parallel processing for inelastic mechanics focused on structural problems. We mention work by Noor et al. (1978); Utku et al. (1982); Storaasli and Bergan (1987) in which they used substructuring to achieve partitions. Fulton and Su (1992) developed techniques to account for different types of elements but used substructures of same element types (non–balanced computations). Hajjar and Abel (1988) developed techniques for dynamic analysis of framed structures with the objective of minimizing communications. Klaas et al. (1994) developed parallel computational techniques for elastic–plastic problems but tied the algorithm to the specific multiprocessor computers used (and specific network connectivity architecture). Farhat (1987) developed the so–called Greedy domain partitioning algorithm but stayed short of using redistribution of domains as a function of developed nonlinearities.

The major parallel programming model in OpenSees (McKenna, 1997) is the so-called Actor model, which is a mathematical model of concurrent computation that has its origins in Hewitt et al. (1973). Actors Agha (1984) are autonomous and concurrently executing objects which execute asynchronously. Actors can create new actors and can send messages to other actors. The Actor model is an Object-Oriented version of message passing in which the Actors represent processes and the methods sent between Actors represent communications.

The Actor model adopts the philosophy that everything is an Actor. This is similar to the everything is an Object philosophy used by object-oriented programming languages, but differs in that object-oriented software is typically executed sequentially, while the Actor model is inherently concurrent, http://en.wikipedia.org/wiki/Actor_model.

An Actor is a computational entity with a behavior such that in response to each message received it can concurrently:

- send a finite number of messages to (other) Actors;
- create a finite number of new Actors;
- designate the behavior to be used for the next message received.

Note that there is no assumed sequence to above actions and that they could in fact be carried out in parallel.

Communications with other Actors occur asynchronously (i.e. the sending Actor does not wait until the message has been received before proceeding with computation), which is the unblocking behavior.
Messages are sent to specific **Actors**, identified by address (sometimes referred to as the Actor’s “mailing address”). As a result, an **Actor** can only communicate with **Actors** for which it has an address which it might obtain in the following ways:

- The address is in the message received;
- The address is one that the **Actor** already had, i.e. it was already an “acquaintance”;
- The address is for a just created **Actor**.

The **Actor** model is characterized by inherent concurrency of computation within and among **Actors**, dynamic creation of **Actors**, inclusion of **Actor** addresses in messages, and interaction only through direct asynchronous message passing with no restriction on message arrival order.

In order to minimize the changes to the sequential Domain Decomposition design presented in previous sections, McKenna (1997) introduces the **Shadow** class. A **Shadow** object is an object in an **Actor**’s local address space. Each **Shadow** is associated with one **Actor** or multiple **Actors** in the case of an aggregation. The **Shadow** object represents the remote object to the objects in the local **Actor**’s space. The **Shadow** object is responsible for sending an appropriate message to the remote **Actor** or **Actors** if broadcasting. The remote **Actor(s)** will then, if required, return the result to the local **Shadow** object, which in turn replies to the local object. The communication process is shown in Figure 202.10.

![Communication Pattern of Actor-Shadow Models](image)

Figure 202.10: Communication Pattern of Actor-Shadow Models McKenna (1997)

Some other new classes of parallel finite element programming are:

- **Channel** is the bridge through which the **Actors** and **Shadows** can communicate.
- **Address** represents the location of a **Channel** object in the machine space. **Channel** objects
send/receive information to/from other Channel objects, whose locations are given by the Address objects.

- **MovableObject** is an object which can send its state from one actor process to another.

- **ObjectBroker** is an object in a local actor process for creating new objects.

- **MachineBroker** is an object in a local actor process that is responsible for creating remote actor processes at the request of Shadow objects in the same local process.

The relation between these classes is shown in Figure 202.11.
Figure 202.11: Class Diagram for Parallel Finite Element Analysis
202.3.3 Dual-Phase Adaptive Load Balancing

From the Figure 202.7, one can easily identify two computational phases that are fundamental to nonlinear elastic-plastic finite element simulations. One is well known as global level equation solving and the other is local level elemental calculations during which the elemental update happens for each element. In nonlinear elastic-plastic finite element simulations, the local computational phase can be much more expensive than the global equation solving phase due to the presence of complex material models and nonlinearity.

In this chapter, the implementation of proposed PDD algorithm has considered load balancing issues on both elemental level elastic-plastic computations and global level equation solving.

202.3.3.1 Elemental Level Load Balancing

The load balancing operation on constitutive level is built on the foundation of adaptive multilevel graph partitioning algorithm available through ParMETIS.

In this chapter, element-based graph is constructed from the Finite Element mesh on which the graph partitioning algorithm acts on to obtain partitions and/or repartitions. Each element will be assigned a vertex tag for identification.

When two elements at least share a single node, we assign an edge to both vertices because the element graph is deemed to be undirected, which means the edge is equally identified by two vertices without ordering required.

We creatively specify vertex weight to represent elemental level computational load for each vertex (element). In the implementation of this chapter, the vertex weight will be automatically updated as simulation progresses to reflect element computation cost. Performance timing has been added for constitutive update routines and the graph data structure will be refreshed every single iteration.

The last metric used is the vertex size of each vertex which basically contains the information that how much memory each vertex (element) requires in order to reproduce itself to other processes during data distribution. Adaptive load balancing is a multi-objective operation in the sense that both edge cut and data migration cost must be minimized simultaneously. The vertex size exactly describes the size of data that need to be shipped via communication. This metric must be correctly obtained for all available element types in order for the multi-objective load balancing algorithm to ensure the best performance.

202.3.3.2 Equation Solving Load Balancing

Parallel equation solving algorithm falls into two major different categories, direct solver and iterative solver.
Direct solver stems from Gaussian-type elimination and effective elimination tree is determined by the sparsity pattern of the stiffness matrix. Load balancing issue is addressed inherently when forming the elimination tree. Various packages such as SPOOLES and SuperLU provide scalable direct solutions to parallel equation systems. Chapter 110.5 discusses in further details about parallel direct solvers that are available as part of the release of this chapter.

Iterative solver has been the focus of this chapter in the sense that special care has been paid to achieve dynamic load balancing for each partition/repartition. The kernel of project-based iterative solvers is matrix-vector multiply. The issues of how to evenly distribute the stiffness matrix in parallel among different processors and how to reorder the sparse matrix to reduce data communications have been the focus of this chapter.

In order to achieve load balancing for parallel iterative solvers, parallel matrix/vector storage scheme and sparse matrix ordering are key factors. In the implementation of this chapter, even row-distribution of stiffness matrix among processing units is assumed. As shown in Figure 202.12, each processing unit has equal number of rows stored locally. The right hand side of the system is the force vector, which will be replicated for each processing unit. In this way, one can expect fastest matrix-vector multiply with the least amount of data needed to be communicated through network. As matrix-vector multiply is performed in parallel, load balancing issue is related to the number of nonzero numbers of the sparse stiffness matrix, which directly determines how many floating point multiplications are needed. In finite element computations, this nonzero pattern is determined by DOF numbering. Bandwidth reducing numbering scheme, or matrix ordering scheme, such as RCM Dongarra et al. (2003), can effectively lead to a sparse pattern that has similar number of nonzero elements on majority of rows as shown in Figure 202.12.

Finite element method inherently possesses compact support. Off-diagonal data of the stiffness matrix need to be synchronized among different processors. In order to reduce the extra overhead involved, in this chapter, several implementation solutions have been considered.

- **Graph Partitioning Phase.** As stated in previous chapters, minimizing edge-cut is one of the main objectives of the partitioning operation on the element graph. One extra benefit is that the bandwidth of the stiffness matrix will be greatly reduced. The number of nodes that need to be synchronized will be greatly reduced.

- **DOF Numbering Phase.** This phase is to renumber the DOFs of the finite element model after data redistribution in order to make sure contributions from local elements will sit on rows that are stored locally. This is done every time when the data migration is triggered. The idea is to start numbering the DOFs from local elements in Processor 1 to local elements in Processor N.
In this way, when the global matrix is formed, local element stiffness matrix will always become clustered along the diagonal.

### 202.3.4 Object-Oriented Design of PDD

The parallel design of PDD basically follows Main-Follower algorithm structure as shown in Figure 202.13 and MPI has been adopted to facilitate inter-processor communications. The Actor/Shadow model described in previous sections is the used in PDD implementation and does nicely interact with parts of OpenSees framework, which uses **Actor** and **Shadow** classes to facilitate the inter-process communication between the main compute process and tied/follower compute processes.

- **Main Compute Process**

  Main compute process assumes the role to orchestrate the whole computation process. OpenSees uses **tcl** as an interpreter (or any other interpreted language that can be embedded into c or C++) to read input scripts from user. In parallel implementation of described here, main compute process
is responsible for establishing the whole model for analysis and then distributing data among subprocessors. An important improvement in this chapter is that the main compute process does not actually create all finite element objects, whose memory space will only be allocated after they are sent to subdomains. This design helps avoid the high memory requirement on the main compute process side. Initial partitioning is done solely by main compute process or in parallel by all working processes. Data movement is coordinated by the main compute process process, in which a complete element graph is kept intact.

As for repartitioning, the main compute process is still responsible for issuing commands to migrate data from this subprocessor to another even though the data is not in main compute process.

- **Follower Compute Process**

The actor model has been used and modifications have been added to avoid unnecessary data communications. Basically speaking, actors in follower compute processes will be waiting for orders until main compute process issues one and then do corresponding work on their own copy of data. The original design in OpenSees framework has disabled follower compute process to initiate communication, which means in order for a sub-processor to communicate with another sub-processor,
it has to send all the data back to the main compute process first. This is highly inefficient and needed to be redeveloped, improved. In this research actor model has been implemented to enable direct communications between sub-processes and this improvement greatly reduced unnecessary communications.

All of the class designs for sequential version of OpenSees can be reused in parallel version following the Object-Oriented paradigm. There are some very important additions however in order to facilitate main–follower parallel processing. In this section, these classes will be revisited and updated/changes/improvements originally developed during this research will be explained thereafter.

- **PartitionedDomain**
  The **PartitionedDomain** class basically inherits all functionality from the **Domain** class in sequential version. This class acts as a container class in the main compute process. It differs from **Domain** class in the respect that all actions performed on the domain will be propagated to all subdomains when doing parallel processing.

- **Subdomain**
  The **Subdomain** is a child class of **Domain**. This class will be instanced by each follower compute process and it covers all functionality of the **Domain** class in sequential version. It can be called as an instance of **Domain** taking care of components only for the local follower compute process.

- **ActorSubdomain & ShadowSubdomain**
  The **Actor/Subdomain** classes are the most important classes for parallel version OpenSees. They are assuming the roles to initiate and facilitate all communications between main and follower compute processes. Both **Actor/Subdomain** will be instanced automatically when user creates follower compute process. 

  **ShadowSubdomain** sits on main compute process. The function of this class is to represent a specific follower compute process in main compute process. Main compute process does not directly interact with follower compute process. Whatever action that needs to be performed by the follower compute process will be issued to **ShadowSubdomain**. This extra layer smooths the communication between main and follower nodes.

  On the other hand, **ActorSubdomain** sits on follower compute process and it hides main compute process from follower compute processes/nodes. All commands from main compute node will be received by **ActorSubdomain** and **ActorSubdomain** will match the command with some actions performed by **Subdomain**.

  **Actor/Subdomain** are extremely important classes in the parallel implementation of this
chapter. They carry all communication functionality required to finish the partition and adaptive repartition.

- **Channel**
  
  Channel is the class that really does the job of sending/receiving data between processors. Only MPI channel has been used in this chapter. Specific data structure, such as ID (integer array), vector (double array) or matrix needs to provide its own implementation for send/receive functionality.

- **FEMObjectBroker**
  
  This class is instanced only at follower compute processors, which is in charge of creating new model data for subdomains. This design isolates model creation from communication classes.

- **Address**
  
  Address class identifies parallel processes. With MPI channel used, the address corresponds to global process ID.

- **DomainPartitioner**
  
  DomainPartitioner assumes the responsibilities of invoking the GraphPartitioner and feeding necessary data to finish the partition/repartition. This class will also be in charge of data migration after partition/repartition is done.

- **SendSelf & RecvSelf**
  
  These two should be called functions rather than classes. SendSelf & RecvSelf are functions implemented to provide copy of model data to finish sending/receiving operations.

The old parallel design of OpenSees is not capable of performing elastic–plastic computations since it was designed and implemented for a single stage loading only. This single stage loading works fine for elastic analysis, but since elastic–plastic materials do have memory, staged loading is essential for any realistic computations with elastic–plastic material. This is particularly true for geotechnical and structural models, where simulations support for staged loading (self weight of soil medium for initial stress, construction process and subsequent static or dynamic loading) is essential if any modeling accuracy is to be achieved. One of new developments in this chapter was the addition of multi-stage elastic-plastic analysis. This improvement included modification of 3D solid and beam elements, Template3Dep/NewTemplate3Dep material models and DRM loading pattern for seismic analysis. Some of the old utility commands, such as “wipeAnalysis”, were improved/redeveloped to enable parallel multi-stage analysis.
The most significant improvement developed during research over the old parallel design of OpenSees is the introduction of load balancing technique by adaptive graph partitioning algorithm through ParMETIS. Major improvements/updates have been introduced in PartitionedDomain, Actor/ShadowSubdomain, DomainPartitioner, FEM_ObjectBroker and Subdomain. Modifications done in this chapter also focus very much on performance issue. In order to reduce unnecessary data communication during partitioning/repartitioning, some functions have been rewritten. The functionality of Actor and ShadowSubdomain have been expanded so that any ActorSubdomain can initiate communication to another ActorSubdomain. The old design of OpenSees had to use main compute process as intermediate layer if subdomains want to exchange information.

For example, if Subdomain No. 1 needs to migrate an Element to Subdomain No. 2, the old design would issue a “remove Element” command from main compute process PartitionedDomain to Subdomain No. 1, then Subdomain No. 1 would remove the Element and send the Element back to main compute process, finally the Element would be migrated to Subdomain No. 2. We can clearly recognize the communication to main compute process is not necessary here. In order to develop adaptive load balancing while minimizing data redistribution cost, the improvement in this chapter is to allow ActorSubdomain at source Subdomain initiates communication with ActorSubdomain at target Subdomains and they can exchange information without recourse to main compute process. So the new communication pattern will be, again for the “migrate element” case, the main compute process will issue an “export element” command to Subdomain No. 1 and a “receive element from Subdomain No. 1” command to Subdomain No. 2, and then the element information will be directly sent from Subdomain No. 1 to No. 2.

Details of implementation are given in following sections.

### 202.3.4.1 MPI_Channel

- Functions `sendnDarray` and `recvnDarray` have been added to facilitate the data communication of Template3D material classes, which are based on `nDarray` tensor data structures.

```cpp
int MPI_Channel::sendnDarray(int,int, const nDarray&, ChannelAddress*)
int MPI_Channel::recvnDarray(int,int, const nDarray&, ChannelAddress*)
```

### 202.3.4.2 MPI_ChannelAddress

- Function `getOtherTag` has been added to get MPI global ID for the specific `MPI_Channel`. This function is mainly used for data migration. It provides the MPI global communicator ID of the target process which the next communication will be directed to.
202.3.4.3 FEM_ObjectBroker

- New functionality to instance 3D continuum brick elements has been added to `getNewElement` function.

```c
Element* FEM_ObjectBroker::getNewElement(EightNodeBrickTag)
```

- New functionality to instance `Template3D/NewTemplate3D` material models for continuum brick elements has been added to `getNewNDMaterial` function.

```c
NDMaterial* FEM_ObjectBroker::getNewNDMaterial(int)
```

- `Template3D` material is a stand-alone material library designed for general elastic-plastic materials. User can define separately `YieldSurface`, `PotentialSurface`, `Scalar Evolution Law` and `Tensorial Evolution Law`. Various material models have been implemented in OpenSees Jeremić and Yang (2002), such as Cam Clay, Drucker Prager and von Mises yield/potential surfaces, Armstrong Frederick nonlinear kinematic hardening law and bounding surface plasticity. All the material models have to be instanced by FEM_ObjectBroker during parallel processing.

```c
YieldSurface* FEM_ObjectBroker::getYieldSurfacePtr(int)
PotentialSurface* FEM_ObjectBroker::getPotentialSurfacePtr(int)
EvolutionLaw_S* FEM_ObjectBroker::getEL_S(int)
EvolutionLaw_T* FEM_ObjectBroker::getEL_T(int)
```

- `NewTemplate3D` material is a newly designed material library which includes more advanced elastic-plastic constitutive models for geomatics, such as Dafalias and Manzari 2004 model. The design of `NewTemplate3D` extends the principle of `Template3D`, in which key parameters describing plasticity model are abstracted as different class objects, such as `YieldFunction`, `PlasticFlow`, etc. In order to reduce unnecessary data allocation, new `MaterialParameter` class has been developed to carry all material parameters. New `ElasticState` has been used to store all intermediate and/or committed stress/strain data. All these material classes have to be instanced by FEM_ObjectBroker during parallel processing and new functions have been implemented in this chapter.

```c
MaterialParameter* FEM_ObjectBroker::getNewMaterialParameterPtr(void)
ElasticState* FEM_ObjectBroker::getNewElasticStatePtr(int)
YieldFunction* FEM_ObjectBroker::getNewYieldFunctionPtr(int)
PlasticFlow* FEM_ObjectBroker::getNewPlasticFlowPtr(int)
```
ScalarEvolution* FEM_ObjectBroker::getNewScalarEvolutionPtr(int)
TensorEvolution* FEM_ObjectBroker::getNewTensorEvolutionPtr(int)

202.3.4.4 Domain

- Timing routines have been added to **update** function to measure computation time of constitutive level iterations for each element during every single loading increment. This metric will be assigned to the corresponding vertex of the element graph as the vertex weight. This metric represents element-level computational load against which subsequent load balancing techniques will be applied.

202.3.4.5 PartitionedDomain

- **addElementalLoad** function has been added to add **ElementalLoad** into **LoadPattern**, which was not supported in the old design.

```cpp
bool PartitionedDomain::addElementalLoad(ElementalLoad*, int)
```

- **repartition** function has been implemented to initiate adaptive repartitioning on the element graph of the **Domain** after every loading increment.

```cpp
int PartitionedDomain::repartition(int)
```

202.3.4.6 Node & DOF_Group

- **sendSelf** and **recvSelf** functions for **Node** class have been changed mainly to deal with the **DOF_Group** object associated with the **Node**. In the old design of parallel version of OpenSees, only one-step static domain partitioning would be invoked so that there is no need to pass the **DOF_Group**. But in this chapter, adaptive load balancing is developed to achieve better performance. The **Node** class should keep the information of its own **DOF_Group**, which guarantees the consistency of the **DOF_Graph** of the whole **Domain**. This point is extremely important when user tries to invoke **Transformation** constraint handler on the **DOF_Graph**. The addition of this feature in **Node** improved the robustness of the whole program.

- **DOF_Group** is a class carries information about the **DOF_Graph** of the analysis model, which will be used to finish assembling the stiff/mass/damping matrices. Each **Node** has its own **DOF_Group** to record the IDs of degree of freedoms in the global analysis model. Function **unSetMyNode** has been introduced to avoid segmentation fault. The reason is that after each round of repartitioning, if data movement is required, the **AnalysisModel** will be wiped off but **Nodes** are still in existence.
Introduction of `unSetMyNode` function separates `Node` from its `DOF_Group` so the `DOF_Group` can be wiped and regenerated for the new model.

### 202.3.4.7 DomainPartitioner

`DomainPartitioner` is one of the most extensively changed classes in this chapter. This class acts as the entry point for `PartitionedDomain` to do domain decomposition and it basically has been rewritten to introduce new partition/repartition functionality and new data structures.

- Function `repartition` is implemented to do repartitioning after each loading increment. Partition and repartition are both implemented in parallel through ParMETIS library in this chapter. This function will collect `ElementGraph` from each `Subdomain` and pass them to `GraphPartitioner`. The global `ElementGraph` will be kept intact from which connectivity/adjacency information will be gathered to assemble child `ElementGraphs` and provide initial graph distribution data for repartition routines. After repartitioning by ParMETIS finishes, the function will verify the new partition against the original one to see if data redistribution is required to achieve load balancing. This `repartition` function also acts as a commander to control the data migration for adaptive load balancing. It issues commands to `ShadowSubdomain` to export/import `Nodes`, `Elements`, `Constraints`, `Loads`, etc.

- The old design of OpenSees used multiplication of prime numbers as index number to record which partitions a specific node belongs to. This is a very good idea because with this approach, we only need one integer for each node to keep track of node partitions, which can be called as an index number for the node. The idea was to name each `Subdomain` with one specific prime number, if a node belongs to this `Subdomain`, we would multiply the index number of the node with the prime number of this `Subdomain`. In order to determine if a node belongs to on specific `Subdomain`, all we need is to divide the index number of the node with the prime number the `Subdomain` represents to see if we can get zero residual.

- The drawback of the old data structure based on prime numbers is that it only works when the number of processing units is small, say less than 16. In 3D continuum models, a single node might belong to up to 8 partitions simultaneously, which happens when a corner node sits on intersections of different `Subdomains`. As we know, prime numbers grow up very fast, multiplication with 8 prime numbers can easily overflow the index number of the node. A new data structure inspired by the Compressed Sparse Row (CSR) storage format popular in sparse matrix calculations has been introduced into in this chapter to solve the problem. One integer array has been used to store the partition data of all nodes, i.e. which partitions this node belongs to. Another integer
array has been employed to record the count of partitions for each node. With these two arrays, we can load as many partitions as we want in our parallel processing.

202.3.4.8 Shadow/ActorSubdomain

As mentioned in previous sections of this chapter, Shadow/ActorSubdomain are the most important classes in parallel design of OpenSees McKenna (1997). ShadowSubdomains represent Subdomains in the main compute process PartitionedDomain. If PartitionedDomain requires one specific Subdomain to carry out some operations, it will send out orders to the ShadowSubdomain associated with the target Subdomain. Then the ShadowSubdomain sets up communication channel to communicate with the Subdomain through ActorSubdomain. ActorSubdomain, on the other hand, sits on each child process as an agent receiving and processing incoming operation requests. The major improvements in this chapter include new functionality for adaptive repartitioning and data migration, and several other minor changes to reduce unnecessary data communications, such as when the Subdomain is required to removeElement, the new design won’t send the element information out, etc. New features will be introduced in this section.

- **ShadowActorSubdomain_Partition**
  New design used ParMETIS to do parallel graph partitioning instead of sequential partitioning by METIS in old design. This improvement helps to reduce partition/repartition overhead and enable the parallel adaptive repartitioning for PDD algorithms proposed in this chapter.

- **ShadowActorSubdomain_BeElmentGraph**
  In order to provide input graphs for adaptive load balancing, all Subdomains have to construct their own subElementGraph, which will be fed into ParMETIS routines for repartitioning.

- **ShadowActorSubdomain_Repartition**
  The repartitioning is implemented in parallel in this chapter so this entry point is set in the ActorSubdomain for each Subdomain.

- **ShadowActorSubdomain_reDistributeData**
  If data migration is needed to achieve load balancing, the main compute process will orchestrate the data redistribution process and the functionality here helps to facilitate the data communications between processes. This is one of the major additions to the existing design. Starting from this point, the ActorSubdomain is able to handle all required data movement on its own and ActorSubdomains representing other Subdomains will connect to the current working ActorSubdomain to receive/send data. Logically only one ActorSubdomain will be doing Shad-
owActorSubdomain_reDistributeData while others including the main compute process will be listening to separate MPI port for data migration requests.

- **ShadowActorSubdomain_recvChangedNodeList**
  This function is used to simplify the data migration routine. With this function, only Nodes/Elements and their associated Constraints, Loads etc. need to be moved between processors.

- **ShadowActorSubdomain_changeMPIChannel**
  This function prepares the current ActorSubdomain for messages from some specific processes. It changes the destination/source for subsequent outgoing/incoming communications, which helps redistributing data after load balancing.

- **ShadowActorSubdomain_restoreChannel**
  The default communication pattern in the old design of OpenSees was one to one, main to follower computer processes. This function helps restoring communication patterns of the whole model after data redistribution finishes.

- **ShadowActorSubdomain_swapNodeFromInternalToExternal**
  Nodes that only belong to one single Subdomain is called internal nodes whose information will be stored only in that specific Subdomain. While for those nodes that belong to more than one Subdomains, their information should be accessible from all Subdomains with which the nodes are associated. Those nodes are called external nodes instead. It is possible that former internal nodes to one Subdomain become external after the adaptive repartitioning. What the old design would do is to remove the internal nodes from that Subdomain, gather the information back to the main compute process and then distribute it externally among those Subdomains as indicated by the newly obtained partitions. The improvement in this chapter avoids unnecessary data communication between current working Subdomain and the main compute process Domain. We can just swap the node in working Subdomain from internal status to external status and then export them to other specified Subdomains. This new design can improve performance if the data migration is extensive by avoiding unnecessary communications.

- **ShadowActorSubdomain_swapNodeFromExternalToInternal**
  This function is introduced due to the same reason as described previously although now the swapping direction is in reverse. It is noted that along with the swapping, removing operations must be invoked for those Subdomains that does not contain the node anymore.

- **ShadowActorSubdomain_exportInternalNode**
  This function handles the situation when a Node does not belong to the current Subdomain
after adaptive repartitioning. The node will be removed from current Subdomain and exported to other Subdomains specified by the graph repartitioning. This again avoids the unnecessary data communication to/from main compute process by directly sending data to other Subdomains.

- ShadowActorSubdomain.resetRecorders

The Recorder have to be reset after data migration to reflect component changes in each Subdomain.

202.3.4.9 Send/RecvSelf

As stated in previous sections, Send/RecvSelf must be provided by all domain components to finish data communication operations, such as Nodes, Elements, Loads, Constraints, Materials etc. In this chapter, new communication functions have been developed for EightNodeBrick element, ElasticIsotropic3D material, Template3Dep/NewTemplate3Dep material. The basic requirement to implement Send/RecvSelf is to replicate the source object instance in target process. For the old design, only one-step initial partitioning is performed and thus greatly simplifies the Send/RecvSelf routines because all the analysis-related information is null or void and only geometry-related data need to be transferred. But in this chapter, data migration is needed periodically to achieve load balance so the Send/RecvSelf has to be redesigned to carry analysis-related information besides the geometry model data. This is extremely important for Element and Material classes because they contain intermediate iteration/solution data of nonlinear finite element simulations. Figure 202.15 shows the class diagrams of brick Element and the associated Template3Dep material model. Send/RecvSelf operations have been implemented also for all classes associated with Template3Dep which are necessary to define a complete material model, such as Cam Clay, Drucker Prager and von Mises PotentialSurfaces, Cam Clay, Drucker Prager and von Mises YieldSurfaces, linear and nonlinear isotropic and kinematic hardening rules, etc.

202.3.5 Graph Partitioning

Graph partitioning approach has been extensively used in implementing domain decomposition type parallel finite element method. The element-based graph naturally becomes the favorite due to the fact that elemental operation forms the fundamental calculation unit in finite element analysis.

In this chapter, element graph has been constructed upon which graph partitioning algorithm acts to get domain decomposition for parallel finite element analysis. In the current implementation of this chapter, vertices of the element graph represent elements of the analysis model. Vertex weight is then specified as the computational load of each element. In elastic-plastic finite element simulations,
the most expensive part has shown to be the elemental level calculations, which include constitutive-level stress update (strain-driven constitutive driver assumed) and formulation of elastic-plastic modulus (or so-called tangent stiffness tensor/matrix). In this research, the wall clock time used by elemental calculations has been dynamically collected and specified as the corresponding vertex weight for each element. The elemental calculation time clearly tells whether the element is elastic or plastified. With this timing metric, the graph can effectively reflect load distribution among elements thus load balancing repartition can be triggered on the graph to redistribute element between processors to achieve more balanced elastic-plastic calculation.

On the other hand, vertex size has to be defined for repartitioning problem as mentioned in previous sections. In this research, vertex size has been specified to be redistribution cost associated with each element. This information depends on the parallel implementation of the software and is discussed in the section immediately following.

### 202.3.5.1 Construction of Element Graph

Each element is considered as one vertex in the element graph. An edge is formed when two elements share a common node. In this chapter, the graph structure is assumed to be undirected, which means the same edge will be added to both vertex ends. The edge is weightless in our application considering the fact that the purpose of minimizing edge-cut is to reduce the data migration when assembling global stiffness matrix. In that sense, the edge of element graph should carry the same weight or, more directly no weight at all.

### 202.3.5.2 Interface to ParMETIS/METIS

Interfaces to both ParMETIS and METIS have been implemented in this chapter. ParMETIS is the parallel implementation of METIS and new adaptive repartitioning functionality is only available through ParMETIS.

All of the graph routines in ParMETIS/METIS take as input the adjacency structure of the graph, the weights of the vertices and edges (if any), and an array describing how the graph is distributed among the processors Karypis et al. (2003). The structure of the graph is represented by the compressed storage format (CSR), extended for the context of parallel distributed-memory computing. We will first describe the CSR format for serial graphs and then describe how it has been extended for storing graphs that are distributed among processors.

- **Serial CSR Format** The CSR format is a widely-used scheme for storing sparse graphs. Here, the adjacency structure of a graph is represented by two arrays, $xadj$ and $adjncy$. Weights on the
vertices and edges (if any) are represented by using two additional arrays, \( vwgt \) and \( adjwgt \). For example, consider a graph with \( n \) vertices and \( m \) edges. In the CSR format, this graph can be described using arrays of the following sizes:

\[
xadj[n + 1], \ vwgt[n], \ adjncy[2m], \ \text{and} \ adjwgt[2m]
\]  

(202.1)

Note that the reason both \( adjncy \) and \( adjwgt \) are of size \( 2m \) is because every edge is listed twice (i.e., as \( (v,u) \) and \( (u,v) \)). Also note that in the case in which the graph is unweighted (i.e., all vertices and/or edges have the same weight), then either or both of the arrays \( vwgt \) and \( adjwgt \) can be set to \( NULL \). ParMETIS_V3_AdaptiveRepart additionally requires a \( vsize \) array. This array is similar to the \( vwgt \) array, except that instead of describing the amount of work that is associated with each vertex, it describes the amount of memory that is associated with each vertex.

The adjacency structure of the graph is stored as follows. Assuming that vertex numbering starts from 0 (C style), the adjacency list of vertex \( i \) is stored in array \( adjncy \) starting at index \( xadj[i] \) and ending at (but not including) index \( xadj[i + 1] \) (in other words, \( adjncy[xadj[i]] \) up through and including \( adjncy[xadj[i + 1] - 1] \)). Hence, the adjacency lists for each vertex are stored consecutively in the array \( adjncy \). The array \( xadj \) is used to point to where the list for each specific vertex begins and ends. Figure 202.14(a) illustrates the CSR format for the 15-vertex graph shown in Figure 202.14(b). If the graph has weights on the vertices, then \( vwgt[i] \) is used to store the weight of vertex \( i \). Similarly, if the graph has weights on the edges, then the weight of edge \( adjncy[j] \) is stored in \( adjwgt[j] \). This is the format that is used by (serial) METIS library routines.

- **Distributed CSR Format** ParMETIS uses an extension of the CSR format that allows the vertices of the graph and their adjacency lists to be distributed among the processors. In particular, PARMETIS assumes that each processor \( P_i \) stores \( n_i \) consecutive vertices of the graph and the corresponding \( m_i \) edges, so that \( n = \sum_i n_i \) , and \( 2m = \sum_i m_i \). Here, each processor stores its local part of the graph in the four arrays \( xadj[n_i + 1], vwgt[n_i], adjncy[m_i], \) and \( adjwgt[m_i] \), using the CSR storage scheme. Again, if the graph is unweighted, the arrays \( vwgt \) and \( adjwgt \) can be set to \( NULL \). The straightforward way to distribute the graph for PARMETIS is to take \( n/p \) consecutive adjacency lists from \( adjncy \) and store them on consecutive processors (where \( p \) is the number of processors). In addition, each processor needs its local \( xadj \) array to point to where each of its local vertices' adjacency lists begin and end. Thus, if we take all the local \( adjncy \) arrays and concatenate them, we will get exactly the same \( adjncy \) array that is used in the serial CSR. However, concatenating the local \( xadj \) arrays will not give us the serial \( xadj \) array. This is because the entries in each local \( xadj \) must point to their local \( adjncy \) array, and so, \( xadj[0] \)
Figure 202.14: An example of the parameters passed to PARMETIS in a three processor case Karypis et al. (2003).

is zero for all processors. In addition to these four arrays, each processor also requires the array $vtxdist[p + 1]$ that indicates the range of vertices that are local to each processor. In particular, processor $P_i$ stores the vertices from $vtxdist[i]$ up to (but not including) vertex $vtxdist[i + 1]$.

Figure 202.14(c) illustrates the distributed CSR format by an example on a three-processor system. The 15-vertex graph in Figure 202.14(a) is distributed among the processors so that each processor gets 5 vertices and their corresponding adjacency lists. That is, Processor Zero gets vertices 0 through 4, Processor One gets vertices 5 through 9, and Processor Two gets vertices 10 through 14. This figure shows the $xadj$, $adjncy$, and $vtxdist$ arrays for each processor. Note that the $vtxdist$ array will always be identical for every processor. All five arrays that describe the distributed CSR format are defined in PARMETIS to be of type $idxtype$. By default $idxtype$ is set to be equivalent...
to type \texttt{int} (i.e., integers). However, \texttt{idxtype} can be made to be equivalent to a \texttt{short int} for certain architectures that use 64-bit integers by default. (Note that doing so will cut the memory usage and communication time required approximately in half.) The conversion of \texttt{idxtype} from \texttt{int} to \texttt{short} can be done by modifying the file \texttt{parmetis.h}. (Instructions are included there.) The same \texttt{idxtype} is used for the arrays that store the computed partitioning and permutation vectors.

When multiple vertex weights are used for multi-constraint partitioning, the $c$ vertex weights for each vertex are stored contiguously in the $vwgt$ array. In this case, the $vwgt$ array is of size $nc$, where $n$ is the number of locally stored vertices and $c$ is the number of vertex weights (and also the number of balance constraints).

New \texttt{GraphPartitioner} class \texttt{ParMETIS} has been developed in this chapter to provide seamless interface to adaptive partitioning/repartitioning routines.

### 202.3.6 Data Redistribution

Data redistribution after repartitioning has been a challenging problem which needs careful study to guarantee correctness of subsequent analysis. In this research, Object-Oriented philosophy has been followed to abstract container classes to facilitate analysis and model data redistribution after repartition.

As for the initial partitioning, only model data, such as geometry parameters, has to be exported to sub-processors, while in adaptive repartitioning finite element simulation, analysis data has to be moved as well. It is extremely important to have well-designed container classes to carry data around. Basic units of finite element analysis, such as nodes and elements naturally become our first choices. Not to give up generality, the design in OpenSees adopts basic iterative approach for nonlinear finite element analysis \cite{Crisfield1997}, important intermediate analysis data include trial data, commit data, incremental data, element residual, element tangent stiffness, etc. Vertex size of each element has been defined as the total number of bytes that have to be transferred between sub-processors.

1. **Node**

   Other than geometric data such as node coordinates and number of degree of freedoms, the \texttt{Node} class contains nodal displacement data which should be sent together with the node to preserve continuity of the analysis model.

2. **Element**

   \texttt{Element} class is the basic construction unit in finite element model. In the design of this research, \texttt{Element} class keeps internal links to \texttt{Template3D} material class \cite{Jeremić2002}. In order to facilitate elastic-plastic simulation, \texttt{EPState} class is constructed to hold all the intermediate response data. This object-oriented abstraction greatly systematize the communication.
pattern. The information on class design is shown in the class diagram Figure 202.15 by Rational Rose Boggs and Boggs (2002).
Figure 202.15: Class Diagram: Major Container Classes for Data Redistribution
All data communication operations have been implemented through the standard `Send/RecvSelf` interface, which forms a complete set of consistent point-to-point communication patterns and is convenient for future additions of new element/materials.
Chapter 203

Library Centric Software Platform Design

203.1 Chapter Summary and Highlights

Veldhuizen (2005); Stroustrup (2005); Ramey (2005); Veldhuizen (2005);

203.1.1 Finite Elements

203.1.1.1 Single Phase Solid Elements

8 Node Brick

20 Node Brick

27 Node Brick

8-20 Node Brick

203.1.1.2 Fully Coupled, Two-Phase (Porous Solid – Pore Fluid) Solid Elements

u-p-U

u-p

203.1.1.3 Structural Elements

Truss

Beam

203.1.1.4 Special Elements

Contact Element

Seismic Isolator Element
203.1.2 Constitutive Integration and Material Models

203.1.2.1 Explicit Integration

203.1.2.2 Implicit Integration

203.1.2.3 Material Models

203.1.3 Modified OpenSees Services Library

PDD...
Chapter 204

Application Programming Interface

(In collaboration with Dr. Nima Tafazzoli and Dr. Yuan Feng)
204.1 Chapter Summary and Highlights

204.2 Introduction

Bloch (2005);
Dmitriev (2004); Stroustrup (2005); Niebler (2005); Mernik et al. (2005); Ward (2003);

204.3 Application Programming Interface for Domain Specific Language (DSL)

204.3.1 Modeling

Start new loading stage:

```c
int start_new_stage(string CurrentStageName);
```

```c
define_model_name(string theModelName)
```

```c
obtain_pseudo_time()
```

```c
wipe_model()
```

```c
check_mesh(string outputfilename)
```


204.3.1.1 Modeling: Material Models

```c
int add_constitutive_model_NDMaterial_linear_elastic_isotropic_3d(int MaterialNumber,
                  double ElasticModulus,
                  double nu,
                  double rho)
```

*MaterialNumber*: Number of the predefined ND material to be used;
*ElasticModulus*: elastic modulus;
*nu*: Poisson’s ratio;
*rho*: density;

```c
add_constitutive_model_NDMaterial_linear_elastic_crossanisotropic(int MaterialNumber,
                  double Ehp,
                  double Evp,
                  double nuhvp,
                  double nuhhp,
                  double Ghvp,
                  double rhop)
```

*MaterialNumber*: Number of the ND material to be used; *Ehp*: Elastic modulus on “horizontal” direction; *Evp*: Elastic modulus on “vertical” direction; *nuhvp*: Poisson ratio for “horizontal” - “vertical” direction; *nuhhp*: Poisson ratio for “horizontal” - “horizontal” direction; *Ghvp*: Shear moduls for “horizontal” - “vertical” direction; *rhop*: density;

```c
add_constitutive_model_NDMaterial_vonmises_perfectly_plastic(int MaterialNumber,
                  int Algorithm,
                  double rho,
                  double E,
                  double v,
                  double k,
```

```
```
double initialconfiningstress,
int number_of_subincrements,
int maximum_number_of_iterations,
double tolerance_1,
double tolerance_2)

MaterialNumber: Number of the ND material to be used; Algorithm: Explicit (=0) or Implicit (=1); rho: density; E: Elastic modulus; v: Poisson’s ratio; k: initial radius of von Mises cylinder; initialconfiningstress: initial confining pressure (positive for compression);

add_constitutive_model_NDMaterial_vonmises_isotropic_hardening(int MaterialNumber,
int Algorithm,
double rho,
double E,
double v,
double k,
double H,
double initialconfiningstress,
int number_of_subincrements,
int maximum_number_of_iterations,
double tolerance_1,
double tolerance_2)

MaterialNumber: Number of the ND material to be used; Algorithm: Explicit (=0) or Implicit (=1); rho: density; E: Elastic modulus; v: Poisson’s ratio; k: initial radius of von Mises cylinder; H: rate of isotropic hardening; initialconfiningstress: initial confining pressure (positive for compression);

add_constitutive_model_NDMaterial_vonmises_kinematic_hardening(int MaterialNumber,
int Algorithm,
double rho,
double E,
double v,
double k,
  double ha,
  double Cr,
  double initialconfiningstress,
  int number_of_subincrements,
  int maximum_number_of_iterations,
  double tolerance_1,
  double tolerance_2)

  \textit{MaterialNumber}: Number of the ND material to be used ; \textit{Algorithm}: Explicit (=0) or Implicit (=1) ; \textit{rho}: density ; \textit{E}: Elastic modulus ; \textit{v}: Poisson’s ratio ; \textit{k}: initial radius of von Mises cylinder ; \textit{ha}: Armstrong-Frederick nonlinear kinematic hardening constant, initial slope ; \textit{Cr}: Armstrong-Frederick nonlinear kinematic hardening constant, asymptote ; \textit{initialconfiningstress}: initial confining pressure (positive for compression);

\begin{verbatim}
add_constitutive_model_NDMaterial_vonmises_linear_kinematic_hardening(int MaterialNumber,
  int Algorithm,
  double rho,
  double E,
  double v,
  double k,
  double H,
  double initialconfiningstress,
  int number_of_subincrements,
  int maximum_number_of_iterations,
  double tolerance_1,
  double tolerance_2)

  MaterialNumber: Number of the ND material to be used ; Algorithm: Explicit (=0) or Implicit (=1) ; rho: density ; E: Elastic modulus ; v: Poisson’s ratio ; k: initial radius of von Mises cylinder ; H: Kinematic hardening rate; initialconfiningstress: initial confining pressure (positive for compression);
\end{verbatim}
add_constitutive_model_NDMaterial_druckerprager_perfectly_plastic(int MaterialNumber,
    int Algorithm,
    double rho,
    double E,
    double v,
    double k,
    double initialconfiningstress,
    int number_of_subincrements,
    int maximum_number_of_iterations,
    double tolerance_1,
    double tolerance_2)

    MaterialNumber: number/Number of the nD material to be used ; AlgorithmType: Explicit (=0)
    or Implicit (=1) ; rho: density ; E: Elastic modulus ; v: Poisson’s ratio ; k: initial equivalent friction
    angle ; initialconfiningstress: initial confining pressure (positive for compression) ;

add_constitutive_model_NDMaterial_druckerprager_isotropic_hardening(int MaterialNumber,
    int Algorithm,
    double rho,
    double E,
    double v,
    double k,
    double H,
    double initialconfiningstress,
    int number_of_subincrements,
    int maximum_number_of_iterations,
    double tolerance_1,
    double tolerance_2)

    MaterialNumber: number/Number of the nD material to be used ; AlgorithmType: Explicit (=0)
    or Implicit (=1) ; rho: density ; E: Elastic modulus ; v: Poisson’s ratio ; k: initial equivalent friction
    angle ; H: rate of isotropic hardening ; initialconfiningstress: initial confining pressure (positive
    for compression) ;
add_constitutive_model_NDMaterial_druckerprager_kinematic_hardening(int MaterialNumber,
    int Algorithm,
    double rho,
    double E,
    double v,
    double k,
    double ha,
    double Cr,
    double initialconfiningstress,
    int number_of_subincrements,
    int maximum_number_of_iterations,
    double tolerance_1,
    double tolerance_2)

MaterialNumber: number/Number of the ND material to be used; Algorithm: Explicit (=0) or Implicit (=1); rho: density; E: Elastic modulus; v: Poisson’s ratio; k: initial equivalent friction angle; ha: Armstrong-Frederick nonlinear kinematic hardening constant, initial slope; Cr: Armstrong-Frederick nonlinear kinematic hardening constant, asymptote; initialconfiningstress: initial confining pressure (positive for compression);

add_constitutive_model_NDMaterial_camclay(int MaterialNumber,
    int Algorithm,
    double rho,
    double e0,
    double M,
    double lambda,
    double kappa,
    double v,
    double Kc,
    double P0,
    double initialconfiningstress,
int number_of_subincrements,
int maximum_number_of_iterations,
double tolerance_1,
double tolerance_2)

MaterialNumber: number/Number of the material to be used; AlgorithmType: Explicit (=0) or Implicit (=1); rho: density; e0: initial void ratio; M: slope of the critical state line; lambda: slope of the Normal Consolidation Line (NCL); kappa: slope of the Unloading-Reloading Line (URL); v: Poisson ratio; Kc: Bulk modulus; initialconfiningstress: initial confining stress/pressure (positive for compression);

add_constitutive_model_NDMaterial_sanisand_2004(int MaterialNumber,
    int Algorithm,
    double rho,
    double e0,
    double G0,
    double nu,
    double Pat,
    double p_cut,
    double Mc,
    double c,
    double lambda_c,
    double xi,
    double ec_ref,
    double m,
    double h0,
    double ch,
    double nb,
    double A0,
    double nd,
    double z_max,
    double cz,
    double initialconfiningstress,
int number_of_subincrements,
int maximum_number_of_iterations,
double tolerance_1,
double tolerance_2)

MaterialNumber: number/Number of the nD material to be used ; AlgorithmType: Explicit (=0) or Implicit (=1) ; rho: density ; e0: initial void ratio ; G0: elastic shear modulus (same unit as stress) ; nu: Poisson’s ratio ; Pat: atmospheric pressure ; p_cut: pressure cut-off ratio ; Mc: ; c: tension-compression strength ratio ; lambda_c: parameter for critical state line ; xi: parameter for critical state line ; ec_ref: reference void ratio for critical state line ; e_c = e_c(lambda(p_c/Pat)^xi m: opening of the yield surface ; h0: bounding parameter ; ch: bounding parameter ; nb: bounding parameter ; A0: dilatancy parameter ; nd: dilatancy parameter ; z_max: fabric parameter ; cz: fabric parameter ; initialconfiningstress initial confining pressure (positive for compression) ;
double p_r,
double rho_c,
double theta_c,
double X,
double z_max,
double cz,
double p0,
double p_in,
int number_of_subincrements,
int maximum_number_of_iterations,
double tolerance_1,
double tolerance_2)

MaterialNumber: Number of the ND material to be used ; Algorithm: Explicit (=0) or Implicit (=1) ; rho: density ; e0: initial void ratio at zero strain ; G0: Reference elastic shear modulus (same unit as stress) ; K0: Reference elastic bulk modulus (same unit as stress) ; Pat: atmospherics pressure for critical state line ; k_c: cut-off factor; for $p < k_c P_{at}$, use $p = k_c P_{at}$ for calculation of $G$; (a default value of $k_c = 0.01$ should work fine) ; alpha_cc: critical state stress ratio ; c: tension-compression strength ratio ; lambda: parameter for critical state line ; xi: parameter for critical state line ; ec_ref: reference void for critical state line, ; $e_c = e_r lambda (p_c / P_{at})^{xi}$ ; m: opening of the yield surface ; h0: bounding parameter ; ch: bounding parameter ; nb: bounding parameter ; A0: dilatancy parameter ; nd: dilatancy parameter ; p_r: LCC parameter ; rho_c: LCC parameter ; theta_c: LCC parameter ; X: LCC parameter ; z_max: fabric parameter ; cz: fabric parameter ; p0: yield surface size ; p_in ;

add_constitutive_model_NDMaterial_pisano(int tag,
    double E_in,
    double v_in,
    double M_in,
    double kd_in,
    double xi_in,
    double h_in,
    double m_in,
    double rho_in,
add_constitutive_model_NDMaterial_accelerated_vonmises_perfectly_plastic(int MaterialNumber, 
    double rho, 
    double E, 
    double v, 
    double k, 
    double initialconfiningstress, 
    int maximum_number_of_iterations, 
    double tolerance_1, 
    double tolerance_2)

add_constitutive_model_NDMaterial_accelerated_vonmises_isotropic_hardening(int MaterialNumber, 
    double rho, 
    double E, 
    double v, 
    double k, 
    double H, 
    double initialconfiningstress, 
    int maximum_number_of_iterations, 
    double tolerance_1, 
    double tolerance_2)

add_constitutive_model_NDMaterial_accelerated_vonmises_kinematic_hardening(int MaterialNumber, 
    double rho, 
    double E, 
    double v, 
    double k, 
    double ha, 
    double Cr, 
    double initialconfiningstress,
int maximum_number_of_iterations,
  double tolerance_1,
  double tolerance_2)

add_constitutive_model_NDMaterial_accelerated_vonmises_linear_kinematic_hardening(int MaterialNumber,
  double rho,
  double E,
  double v,
  double k,
  double H,
  double initialconfiningstress,
  int maximum_number_of_iterations,
  double tolerance_1,
  double tolerance_2)

add_constitutive_model_NDMaterial_accelerated_druckerprager_perfectly_plastic(int MaterialNumber,
  double rho,
  double E,
  double v,
  double k,
  double initialconfiningstress,
  int maximum_number_of_iterations,
  double tolerance_1,
  double tolerance_2)

add_constitutive_model_NDMaterial_accelerated_druckerprager_isotropic_hardening(int MaterialNumber,
  double rho,
  double E,
  double v,
double k,
double H,
double initialconfiningstress,
int maximum_number_of_iterations,
double tolerance_1,
double tolerance_2)

add_constitutive_model_NDMaterial_accelerated_druckerprager_kinematic_hardening(int MaterialNumber,
    double rho,
    double E,
    double v,
    double k,
    double ha,
    double Cr,
    double initialconfiningstress,
    int maximum_number_of_iterations,
    double tolerance_1,
    double tolerance_2)

add_constitutive_model_NDMaterial_accelerated_camclay(int MaterialNumber,
    double rho,
    double e0,
    double M,
    double lambda,
    double kappa,
    double v,
    double Kc,
    double p0,
    double initialconfiningstress,
    int maximum_number_of_iterations,
    double tolerance_1,
    double tolerance_2)
add_constitutive_model_uniaxial_elastic(int MaterialNumber,
    double elasticmodulus,
    double eta)

MaterialNumber: unique material Number ;
elasticmodulus: elastic modulus of the material ;
eta: damping tangent ;

add_constitutive_model_uniaxial_concrete02(int MaterialNumber,
    double fpc, double epsc0, double fpcu,
    double epscu, double rat, double ft,
    double Ets)

MaterialNumber: unique material Number ;
fpc: compressive strength ;
epsc0: strain at compressive strength ;
fpcu: crushing strength ;
epscu: strain at crushing strength ;
lambd: ratio between unloading slope at epscu and initial slope ;
ft: tensile strength ;
Ets: tension softening stiffness (absolute value) (slope of the linear tension softening branch) ;

int add_constitutive_model_uniaxial_steel01(int MaterialNumber,
    double fy,
    double Ep,
    double Hd,
    int a1,
    int a2,
    int a3,
    int a4)

MaterialNumber: unique material Number ;
fy: yield strength ;
Ep: initial elastic tangent ;
Hd: strain-hardening ratio (ratio between post-yield tangent and initial elastic tangent) ;
a1, a2, a3, a4: isotropic hardening parameters ;
a1: isotropic hardening parameter, increase of compression yield envelope as proportion of yield strength after a plastic strain of a2*(fy/Ep) . ;
a2: isotropic hardening parameter (see explanation under a1) ;
a3: isotropic hardening parameter, increase of tension yield
envelope as proportion of yield strength after a plastic strain of \( a_4 \times (f_y/E_p) \); \( a_4 \): isotropic hardening parameter (see explanation under \( a_3 \));

```c
int add_constitutive_model_uniaxial_steel02(int MaterialNumber,
    double fy, double E0, double b,
    double R0, double cR1, double cR2,
    double a1, double a2, double a3, double a4)
```

- **MaterialNumber**: unique material object Number; \( f_y \): yield strength; \( E_0 \): initial elastic tangent; \( b \): strain-hardening ratio (ratio between post-yield tangent and initial elastic tangent); \( R_0, cR_1, cR_2 \): control the transition from elastic to plastic branches. Recommended values: \( R_0 = \) between 10 and 20, \( cR_1 = 0.925, cR_2 = 0.15 \); \( a_1, a_2, a_3, a_4 \): isotropic hardening parameters; \( a_1 \): isotropic hardening parameter, increase of compression yield envelope as proportion of yield strength after a plastic strain of \( a_2 \times (F_y/E) \); \( a_2 \): isotropic hardening parameter (see explanation under \( a_1 \)); \( a_3 \): isotropic hardening parameter, increase of tension yield envelope as proportion of yield strength after a plastic strain of \( a_4 \times (F_y/E) \); \( a_4 \): isotropic hardening parameter (see explanation under \( a_3 \));

### 204.3.1.2 Modeling: Nodes

```c
int add_node(int NodeNumber,
    int number_of_DOFs,
    double coordinate_x,
    double coordinate_y,
    double coordinate_z)
```

- **NodeNumber**: integer Number identifying node; **number_of_DOFs**: number of degrees of freedom for node; **coordinate_x**: \( x \) coordinate of the node; **coordinate_y**: \( y \) coordinate of the node; **coordinate_z**: \( z \) coordinate of the node;

```c
int remove_node(int NodeNumber)
```
NodeNumber: integer Number identifying the node to be removed.

```c
int add_mass_to_node(int NodeNumber,
                     double massvalue1,
                     double massvalue2,
                     double massvalue3)
```

```c
int add_mass_to_node(int NodeNumber,
                     double massvalue1,
                     double massvalue2,
                     double massvalue3,
                     double massvalue4,
                     double massvalue5,
                     double massvalue6)
```

```c
int add_mass_to_node(int NodeNumber,
                     double massvalue1,
                     double massvalue2,
                     double massvalue3,
                     double massvalue4,
                     double massvalue5,
                     double massvalue6,
                     double massvalue7)
```

NodeNumber: integer Number of the node that mass would be applied to; massvalue(#): the amount of mass assigned to each degree of freedom.

204.3.1.3 Modeling: Finite Elements

```c
add_element_truss(int ElementNumber,
                   int iNode,
                   int jNode,
                   int MaterialNumber,
                   double sectionarea,
```
double rho)

ElementNumber: unique element object Number; dimension: number of dimensions of the beam; iNode, jNode: end nodes; MaterialNumber: Number of the uniaxial material to be used; sectionarea: section area of the truss element; rho: density;

add_element_beam_elastic(int ElementNumber,
    double A,
    double E,
    double G,
    double Jx,
    double Iy,
    double Iz,
    int iNode,
    int jNode,
    double rho,
    double vecxzPlane_X, double vecxzPlane_Y, double vecxzPlane_Z,
    double jntOffsetI_X, double jntOffsetI_Y, double jntOffsetI_Z,
    double jntOffsetJ_X, double jntOffsetJ_Y, double jntOffsetJ_Z)

ElementNumber: unique element object Number; A: section area; E: Young’s modulus; G: Shear Modulus; Jx: polar moment of inertia; Iy: moment of inertia around y; Iz: moment of inertia around z; iNode, jNode: end nodes; TransformationNumber: identifier for previously-defined coordinate-transformation (CrdTransf) object; rho: density; sectionTag: identifier for previously-defined section object;

add_element_beam_elastic_lumped_mass(int ElementNumber,
    double A,
    double E,
    double G,
    double Jx,
    double Iy,
double Iz,  
int iNode,  
int jNode,  
double rho,  
double vecxzPlane_X, double vecxzPlane_Y, double vecxzPlane_Z,  
double jntOffsetI_X, double jntOffsetI_Y, double jntOffsetI_Z,  
double jntOffsetJ_X, double jntOffsetJ_Y, double jntOffsetJ_Z)  

ElementNumber: unique element object Number ;  
A: section area ;  
E: Young’s modulus ;  
G: Shear Modulus ;  
Jx: polar moment of inertia ;  
Iy: moment of inertia around y ;  
Iz: moment of inertia around z ;  
iNode, jNode: end nodes ;  
TransformationNumber: identifier for previously-defined coordinate-transformation (CrdTransf) object ;  
rho: density ;  
sectionTag: identifier for previously-defined section object ;  

add_element_brick_8node(int ElementNumber,  
            int node_numb_1,  
            int node_numb_2,  
            int node_numb_3,  
            int node_numb_4,  
            int node_numb_5,  
            int node_numb_6,  
            int node_numb_7,  
            int node_numb_8,  
            int MaterialNumber)  

    elementTag: unique element object Number ;  
    node_numb_: eight node numbers specified in appropriate order ;  
    materialTag: material Number associated with previously-defined NDMaterial object ;  

add_element_brick_8node_elastic(int ElementNumber,  
                                    int node_numb_1,  
                                    int node_numb_2,
add_element_brick_20node(int ElementNumber,
                        int node_numb_1,
                        int node_numb_2,
                        int node_numb_3,
                        int node_numb_4,
                        int node_numb_5,
                        int node_numb_6,
                        int node_numb_7,
                        int node_numb_8,
                        int node_numb_9,
                        int node_numb_10,
                        int node_numb_11,
                        int node_numb_12,
                        int node_numb_13,
                        int node_numb_14,
                        int node_numb_15,
                        int node_numb_16,
                        int node_numb_17,
                        int node_numb_18,
                        int node_numb_19,
                        int node_numb_20,
                        int MaterialNumber)

elementTag: unique element object Number; node_numb#: eight node numbers specified in appropriate order; materialTag: material Number associated with previously-defined NDMaterial object;
int MaterialNumber)

ElementNumber: unique element object Number; node_num_#: eight node numbers specified in appropriate order; MaterialNumber: material Number associated with previously-defined NDMaterial object;

add_element_brick_20node_elastic(int ElementNumber,
    int node_num_1,
    int node_num_2,
    int node_num_3,
    int node_num_4,
    int node_num_5,
    int node_num_6,
    int node_num_7,
    int node_num_8,
    int node_num_9,
    int node_num_10,
    int node_num_11,
    int node_num_12,
    int node_num_13,
    int node_num_14,
    int node_num_15,
    int node_num_16,
    int node_num_17,
    int node_num_18,
    int node_num_19,
    int node_num_20,
    int MaterialNumber)

ElementNumber: unique element object Number; node_num_#: eight node numbers specified in appropriate order; MaterialNumber: material Number associated with previously-defined NDMaterial object;
int add_element_brick_27node(int ElementNumber,
                int node_num_1,
                int node_num_2,
                int node_num_3,
                int node_num_4,
                int node_num_5,
                int node_num_6,
                int node_num_7,
                int node_num_8,
                int node_num_9,
                int node_num_10,
                int node_num_11,
                int node_num_12,
                int node_num_13,
                int node_num_14,
                int node_num_15,
                int node_num_16,
                int node_num_17,
                int node_num_18,
                int node_num_19,
                int node_num_20,
                int node_num_21,
                int node_num_22,
                int node_num_23,
                int node_num_24,
                int node_num_25,
                int node_num_26,
                int node_num_27,
                int MaterialNumber)

ElementNumber: unique element object Number; node_num_#: eight node numbers specified in
appropriate order; MaterialNumber: material Number associated with previously-defined NDMaterial
object;
int add_element_brick_27node_elastic(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    int node_numb_5,
    int node_numb_6,
    int node_numb_7,
    int node_numb_8,
    int node_numb_9,
    int node_numb_10,
    int node_numb_11,
    int node_numb_12,
    int node_numb_13,
    int node_numb_14,
    int node_numb_15,
    int node_numb_16,
    int node_numb_17,
    int node_numb_18,
    int node_numb_19,
    int node_numb_20,
    int node_numb_21,
    int node_numb_22,
    int node_numb_23,
    int node_numb_24,
    int node_numb_25,
    int node_numb_26,
    int node_numb_27,
    int MaterialNumber)

ElementNumber: unique element object Number ; node_numb_: eight node numbers specified in appropriate order ; MaterialNumber: material Number associated with previsouly-defined NDMaterial object ;

Jeremić et al., University of California, Davis version: 28. May, 2021, 17:09
add_element_brick_8node_up(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    int node_numb_5,
    int node_numb_6,
    int node_numb_7,
    int node_numb_8,
    int MaterialNumber,
    double porosity,
    double alpha,
    double rho_s,
    double rho_f,
    double k_x,
    double k_y,
    double k_z,
    double K_s,
    double K_f)

add_element_brick_8node_upU(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    int node_numb_5,
    int node_numb_6,
    int node_numb_7,
    int node_numb_8,
    int MaterialNumber,
    double porosity,
    double alpha,
    double rho_s,
double rho_f,
double k_x,
double k_y,
double k_z,
double K_s,
double K_f)

add_element_brick_20node_upU(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    int node_numb_5,
    int node_numb_6,
    int node_numb_7,
    int node_numb_8,
    int node_numb_9,
    int node_numb_10,
    int node_numb_11,
    int node_numb_12,
    int node_numb_13,
    int node_numb_14,
    int node_numb_15,
    int node_numb_16,
    int node_numb_17,
    int node_numb_18,
    int node_numb_19,
    int node_numb_20,
    int MaterialNumber,
    double porosity,
    double alpha,
    double rho_s,
    double rho_f,
double k_x,
    double k_y,
    double k_z,
    double K_s,
    double K_f)

add_element_brick_8node_variable_number_of_gauss_points(int ElementNumber,
    int number_of_gauss_points,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    int node_numb_5,
    int node_numb_6,
    int node_numb_7,
    int node_numb_8,
    int MaterialNumber)

add_element_brick_20node_variable_number_of_gauss_points(int ElementNumber,
    int number_of_gauss_points,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    int node_numb_5,
    int node_numb_6,
    int node_numb_7,
    int node_numb_8,
    int node_numb_9,
    int node_numb_10,
    int node_numb_11,
    int node_numb_12,
int node_numb_13,
int node_numb_14,
int node_numb_15,
int node_numb_16,
int node_numb_17,
int node_numb_18,
int node_numb_19,
int node_numb_20,
int MaterialNumber)

add_element_brick_27node_variable_number_of_gauss_points(int ElementNumber,
    int number_of_gauss_points,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    int node_numb_5,
    int node_numb_6,
    int node_numb_7,
    int node_numb_8,
    int node_numb_9,
    int node_numb_10,
    int node_numb_11,
    int node_numb_12,
    int node_numb_13,
    int node_numb_14,
    int node_numb_15,
    int node_numb_16,
    int node_numb_17,
    int node_numb_18,
    int node_numb_19,
    int node_numb_20,
    int node_numb_21,
int node_numb_22,
int node_numb_23,
int node_numb_24,
int node_numb_25,
int node_numb_26,
int node_numb_27,
int MaterialNumber)

add_element_brick_variable_node_8_to_27(int ElementNumber, int number_of_gauss_points,
  int node_numb_1, int node_numb_2, int node_numb_3
  int node_numb_4, int node_numb_5, int node_numb_6
  int node_numb_7, int node_numb_8, int node_numb_9
  int node_numb_10, int node_numb_11, int node_numb_12
  int node_numb_13, int node_numb_14, int node_numb_15
  int node_numb_16, int node_numb_17, int node_numb_18
  int node_numb_19, int node_numb_20, int node_numb_21
  int node_numb_22, int node_numb_23, int node_numb_24
  int node_numb_25, int node_numb_26, int node_numb_27
  int MaterialNumber)

add_element_contact_3dof_to_3dof(int ElementNumber,
  int iNode,
  int jNode,
  double Knormal,
  double Ktangent,
  double frictionRatio,
  double x_local_1,
  double x_local_2,
  double x_local_3)

add_element_contact_nonlinear_3dof_to_3dof(int ElementNumber,
  int iNode,
int jNode,
double Knormal,
double Ktangent,
double frictionRatio,

double maximum_gap,
double maximum_normal_stiffness,
    double x_local_1,
    double x_local_2,
    double x_local_3)

add_element_contact_nonlinear_3dof_to_7dof(int ElementNumber,
    int iNode,
    int jNode,
    double Knormal,
    double Ktangent,
    double frictionRatio,
    double maximum_gap,
    double maximum_normal_stiffness,
    double x_local_1,
    double x_local_2,
    double x_local_3)

add_element_shell_andes_3node(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    double thickness,
    int MaterialNumber)

add_element_shell_andes_4node(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
int node_numb_3,
int node_numb_4,
double thickness,
int MaterialNumber)

add_element_shell_MITC4(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    double thickness,
    int MaterialNumber)

add_element_shell_NewMITC4(int ElementNumber,
    int node_numb_1,
    int node_numb_2,
    int node_numb_3,
    int node_numb_4,
    double thickness,
    int MaterialNumber)

add_element_penalty(int ElementNumber,
    int node1,
    int node2,
    double penalty_stiffness,
    int dof)

add_element_penalty_for_applying_generalized_displacement(int ElementNumber,
    int Exist_Node,
    double penalty_stiffness,
    int direction)
add_element_rank_one_deficient_elastic_pinned_fixed_beam(int ElementNumber,
        double A,
        double E,
        double G,
        double Jx,
        double Iy,
        double Iz,
        int iNode,
        int jNode,
        double rho,
        double vecxzPlane_X, double vecxzPlane_Y, double vecxzPlane_Z,
        double jntOffsetI_X, double jntOffsetI_Y, double jntOffsetI_Z,
        double jntOffsetJ_X, double jntOffsetJ_Y, double jntOffsetJ_Z)

add_element_beam_displacement_based(int ElementNumber,
        int iNode,
        int jNode,
        int numberofintegrationpoints,
        int SectionNumber,
        double rho,
        string integrationrule,
        double vecInLocXZPlane_x, double vecInLocXZPlane_y, double vecInLocXZPlane_z,
        double rigJntOffset1_x, double rigJntOffset1_y, double rigJntOffset1_z,
        double rigJntOffset2_x, double rigJntOffset2_y, double rigJntOffset2_z)

int remove_element(int ElementNumber)

ElementNumber: number identifying the element to be removed;

204.3.1.4 Modeling: Damping
int add_damping_rayleigh(int dampingNumber,
            double a0,
            double a1,
            string which_stiffness_to_use)

            dampingNumber: damping Number number to be used in element definition ; a0, a1: Rayleigh order
            damping coefficients ; which_stiffness_to_use: Initial_Stiffness/Current_Stiffness/Last_Commited_Stiff

add_damping_caughey3rd(int dampingNumber, double a0, double a1, double a2, string which_stiffness_to_use)

add_damping_caughey4th(int dampingNumber, double a0, double a1, double a2, double a3, string which_stiffness_to_use)

int add_damping_to_element(int elementNumber,
            int dampingNumber)

            dampingNumber: damping number to be assigned to element ; elementNumber: element number
            which damping is going to be assigned to ;

int add_damping_to_node(int nodeNumber,
            int dampingNumber)

            dampingNumber: damping number to be assigned to node (note that only the mass proportional
            coefficient will be used for node) ; nodeNumber: node number which damping is going to be assigned
to ;

204.3.1.5  Modeling: Constraints, Supports, Tied Nodes Connections, etc.

int add_support_to_all_dofs_of_node(int NodeNumber)

            NodeNumber: integer Number of the node to be fixed ;
int add_support_to_node(int NodeNumber, 
    int dof_number)

NodeNumber: integer Number identifying the node to be constrained; dof_number: dof to be fixed;

int add_equaldof_to_two_nodes(int nodeRetain, 
    int nodeConstr, 
    int dofID1, 
    int dofID2, 
    ..., 
    int dofID7)

nodeRetain: integer Number identifying the retained, or master node (rNode); nodeConstr: integer Number identifying the constrained, or slave node (cNode); dofID: nodal degrees-of-freedom that are constrained at the nodeConstr to be the same as those at the nodeRetain. Valid range is from 1 to 7.

int remove_support_from_node_by_fixity_number(int FixityNumber)

FixityNumber: integer Number identifying the fixity to be removed;

int remove_support_from_node(int NodeNumber, int dofNumber)

NodeNumber: integer Number identifying the node number; dofNumber: integer Number identifying the dof number;

int remove_equaldof_from_node(int NodeNumber)

FixityNumber: integer Number identifying the fixity to be removed;
204.3.1.6 Modeling: Static Loads

```c
add_force_time_history_linear(int PatternNumber, 
    int NodeNumber, 
    int dof_to_be_shaken, 
    double final_load_value)
```

```c
add_force_time_history_path_series(int PatternNumber, 
    int NodeNumber, 
    int dof_to_be_shaken, 
    double TimeIncrement, 
    double LoadFactor, 
    string Forceinputfilename)
```

```c
add_force_time_history_path_time_series(int PatternNumber, 
    int NodeNumber, 
    int dof_to_be_shaken, 
    double LoadFactor, 
    string Forceinputfilename)
```

```c
add_load_selfweight_to_element(int SelfWeightNumber, 
    int ElementNumber, int AccelerationFieldNumber)
```

```c
add_acceleration_field(int GravityFieldNumber, 
    double accelerationfield_x, 
    double accelerationfield_y, 
    double accelerationfield_z)
```
add_load_constant_normal_pressure_to_8node_brick_surface(int SurfaceLoadNumber,
        int ElementNumber,
        int Node_1,
        int Node_2,
        int Node_3,
        int Node_4,
        double SurfaceLoadMagnitude)

add_load_different_normal_pressure_to_8node_brick_surface(int SurfaceLoadNumber,
        int ElementNumber,
        int Node_1,
        int Node_2,
        int Node_3,
        int Node_4,
        double SurfaceLoadMagnitude1,
        double SurfaceLoadMagnitude2,
        double SurfaceLoadMagnitude3,
        double SurfaceLoadMagnitude4)

add_load_constant_normal_pressure_to_20node_brick_surface(int SurfaceLoadNumber,
        int ElementNumber,
        int Node_1,
        int Node_2,
        int Node_3,
        int Node_4,
        int Node_5,
        int Node_6,
        int Node_7,
        int Node_8,
        double SurfaceLoadMagnitude)
add_load_different_normal_pressure_to_20node_brick_surface(int SurfaceLoadNumber,
    int ElementNumber,
    int Node_1,
    int Node_2,
    int Node_3,
    int Node_4,
    int Node_5,
    int Node_6,
    int Node_7,
    int Node_8,
    double SurfaceLoadMagnitude1,
    double SurfaceLoadMagnitude2,
    double SurfaceLoadMagnitude3,
    double SurfaceLoadMagnitude4,
    double SurfaceLoadMagnitude5,
    double SurfaceLoadMagnitude6,
    double SurfaceLoadMagnitude7,
    double SurfaceLoadMagnitude8)

add_load_constant_normal_pressure_to_27node_brick_surface(int SurfaceLoadNumber,
    int ElementNumber,
    int Node_1,
    int Node_2,
    int Node_3,
    int Node_4,
    int Node_5,
    int Node_6,
    int Node_7,
    int Node_8,
    int Node_9,
    double SurfaceLoadMagnitude)

add_load_different_normal_pressure_to_27node_brick_surface(int SurfaceLoadNumber,
```c
int ElementNumber,
int Node_1,
int Node_2,
int Node_3,
int Node_4,
int Node_5,
int Node_6,
int Node_7,
int Node_8,
int Node_9,
double SurfaceLoadMagnitude1,
double SurfaceLoadMagnitude2,
double SurfaceLoadMagnitude3,
double SurfaceLoadMagnitude4,
double SurfaceLoadMagnitude5,
double SurfaceLoadMagnitude6,
double SurfaceLoadMagnitude7,
double SurfaceLoadMagnitude8,
double SurfaceLoadMagnitude9)
```

```c
add_penalty_displacement_time_history_linear(int PatternNumber,
                                          int PenaltyElementNumber,
                                          int dof_to_be_shaken,
                                          double Final_Displacement_Value)
```

```c
add_penalty_displacement_time_history_path_series(int PatternNumber,
                                                   int PenaltyElementNumber,
                                                   int dof_to_be_shaken,
                                                   double TimeIncrement,
                                                   double LoadFactor,
                                                   string Displacementinputfilename)
```
add_single_point_constraint_to_node(int NodeNumber,
        int dof_number,
        double DOFvalue)

204.3.1.7 Modeling: Dynamic Loads

add_load_pattern_domain_reduction_method

add_load_pattern_domain_reduction_method(int PatternNumber,
        double dt,
        double factor,
        int numberofsteps,
        int numberofdrmnodes,
        int numberofdrmelements,
        double xpositive,
        double xminus,
        double ypositive,
        double yminus,
        double zpositive,
        double zminus,
        string ElementNumbersFilename,
        string NodeNumbersFilename,
        string DisplacementTimeHistoryFilename,
        string AccelerationTimeHistoryFilename)

Inputs: PatternNumber: number assigned to DRM load pattern; dt: time interval of input files for time histories; factor: factor to multiply to the input time history; numberofsteps: Number of the time steps in acceleration/displacement time history; numberofdrmnodes: Number of the nodes in DRM layer; numberofdrmelements: Number of the elements in DRM layer; xpositive, xminus: boundary layer range in x direction; ypositive, yminus: boundary layer range in y direction; zpositive, zminus: boundary layer range in z direction; ElementNumbersFilename:
File including element numbers inside the plastic bowl (1 element number per line in the input file); NodeNumbersFilename: File including node numbers inside the plastic bowl (1 node number per line in the input file); DisplacementTimeHistoryFilename: File including displacement time history (in each line write the values of displacement in time for first degree of freedom of the first node defined in NodeNumbersFilename, next line should have the values for second dof of the first node and continue for all degrees of freedom. Then move to the second node defined in NodeNumbersFilename and ... ; AccelerationTimeHistoryFilename: File including acceleration time history (in each line write the values of displacement in time for first degree of freedom of the first node defined in NodeNumbersFilename, next line should have the values for second dof of the first node and continue for all degrees of freedom. Then move to the second node defined in NodeNumbersFilename and ... ;

remove_load(int LoadPatternNumber)

add_load_pattern_domain_reduction_method_save_forces(int PatternNumber, double dt, double factor, int numberofsteps, int numberofdrmnodes, int numberofdrmelements, double xpositive, double xminus, double ypositive, double yminus, double zpositive, double zminus, string ElementNumbersFilename, string NodeNumbersFilename, string DisplacementTimeHistoryFilename, string AccelerationTimeHistoryFilename, string ForceTimeHistoryFilename)

add_load_pattern_domain_reduction_method_restore_forces(int PatternNumber,
double dt,
double factor,
int numberOfSteps,
int numberOfDrmmNodes,
int numberOfDrmeElements,
double xPositive,
double xMinus,
double yPositive,
double yMinus,
double zPositive,
double zMinus,
string ElementNumbersFilename,
string NodeNumbersFilename,
string ForceTimeHistoryFilename)

204.3.1.8 Modeling: Prescribed Displacements

add_imposed_motion(int GroundMotionNumber,
    int NodeNumber,
    int degree_of_freedom,
    double timestep,
    double displacement_scale,
    string displacementfilename,
    double velocity_scale,
    string velocityfilename,
    double acceleration_scale,
    string accelerationfilename)

204.3.1.9 Solid-Fluid Interface

Two new APIs and corresponding DSL commands have been added to RealESSI.

- define solid fluid interface “interface_name”
  This API is used to define solid fluid interface. Passing parameters into RealESSI to initialize our interface class SSFI.
• simulate No. steps using solid fluid interaction transient algorithm \( \text{time\_step} = \langle \text{time} \rangle \)

This API aims to launch solid fluid transient interaction analysis. The time step defined here refers to the time step for the transient analysis in solid domain. The time step for fluid domain can be different and defined in the input files for OpenFOAM.

204.3.1.10 Outputs to mySQL database

```c
restore_response_of_model_mysql_format(int Node_Number, int DOF_Number, int Step_Number,
                                        string databaseName, string host,
                                        string username, string password, unsigned int port,
                                        string socket)
```

```c
restore_state_of_model_mysql_format(string databaseName, string host,
                                     string username, string password, unsigned int port,
                                     string socket)
```

```c
save_response_of_model_mysql_format(string databaseName, string host,
                                     string username, string password, unsigned int port,
                                     string socket)
```

```c
save_state_of_model_mysql_format(string databaseName, string host,
                                  string username, string password, unsigned int port,
                                  string socket)
```

204.3.2 Simulation

204.3.2.1 Simulation: Solvers

Definition of system of equation (linear) solvers to be used.

```c
int define_solver_profilespd_for_analysis()
```
204.3.2.2 Simulation: Static Solution Advancement

Definition of static solution advancement algorithms (see more in section 107.6 on page 495).

```c
int define_static_solution_advancement_integrator_displacement_control(int node_number,
    int doftomove,
    double dispincrement)
```

`dispincrement`: increment of displacement in each step of analysis; `node_number`: node whose response controls the solution; `dof` controls the solution. Valid range is from 1 through the number of nodal degrees-of-freedom.

```c
int define_static_solution_advancement_integrator_load_control(double loadstep)
```

`loadstep`: load step size;

204.3.2.3 Simulation: Dynamic Solution Advancement

Definition of dynamic, time integration/advancement algorithms (see more in section 108.3 on page 507).

```c
int define_dynamic_solution_advancement_integrator_hilber_hughes_taylor(double HHT_Alpha)
```

`HHT_Alpha`: HHT $\alpha$ parameter;

```c
int define_dynamic_solution_advancement_integrator_newmark(double gamma, double beta)
```

`newmark_gamma`, `newmark_beta`: Newmark $\gamma$ and $\beta$ parameters;
204.3.2.4 Simulation: Solution Algorithms

Definition of solution algorithms to be used:

```c
int define_algorithm_with_no_convergence_check_for_analysis()
```

```c
int define_algorithm_newton_for_analysis()
```

```c
int define_algorithm_modifiednewton_for_analysis()
```

204.3.2.5 Simulation: Convergence Criteria

```c
int define_convergence_test_energyincrement_for_analysis(double theTol,
                          int maxIter,
                          int PrintFlag)
```

- `theTol`: convergence tolerance; `maxIter`: maximum number of iterations that will be performed before "failure to converge" is returned; `PrintFlag`: flag used to print information on convergence (optional): 0: no print output; 1: print information on each step; 2: print information when convergence has been achieved; 4: print norm, dU and dR vectors; 5: at convergence failure, carry on, print error message, but do not stop analysis;

```c
int define_convergence_test_normdisplacementincrement_for_analysis(double theTol,
                                                                         int maxIter,
                                                                         int PrintFlag)
```

- `theTol`: convergence tolerance; `maxIter`: maximum number of iterations that will be performed before "failure to converge" is returned; `PrintFlag`: flag used to print information on convergence (optional): 0: no print output; 1: print information on each step; 2: print information when convergence has been achieved; 4: print norm, dU and dR vectors; 5: at convergence failure, carry on, print error message, but do not stop analysis;
int define_convergence_test_normunbalance_for_analysis(double theTol,
        int maxIter,
        int PrintFlag)

    theTol: convergence tolerance ; maxIter: maximum number of iterations that will be performed
before "failure to converge" is returned ; PrintFlag: flag used to print information on convergence
(optional) ; 0: no print output ; 1: print information on each step ; 2: print information when convergence
has been achieved ; 4: print norm, dU and dR vectors ; 5: at convergence failure, carry on, print error
message, but do not stop analysis ;

204.3.2.6 Simulating Response

int simulate_using_static_multistep(int numSteps)

    numSteps: number of static analysis steps which will advance the solution, ;

int simulate_using_static_onestep()

int simulate_using_transient_multistep(double dT,
        int numSteps)

    dT: time-step increment ; numSteps: number of time steps ;

int simulate_using_transient_onestep()

int simulate_using_transient_variable_multistep(double dT,
        int numSteps,
        double dtMin,
        double dtMax,
        int Jd)


\[ dT: \text{ time-step increment} \; ; \; \text{numSteps}: \text{ number of time steps} \; ; \; \text{dtMin, dtMax}: \text{ minimum and maximum time steps} \; ; \; Jd: \text{ ideal number of iterations performed at each step} \; ; \]

---

```c
int simulate_using_transient_variable_onestep()
```

---

```c
int simulate_using_eigen_analysis(int number_of_eigen_values)
```

### 204.4 Application Programming Interface for Constitutive Simulations

### 204.5 Application Programming Interface for Finite Elements
204.6 Adding New Finite Element into Real-ESSI Simulator

This section illustrates how to add a new element in Real-ESSI simulator. A detailed description of each steps involved is given. The developer is expected to understand these steps and replicate it for their new element. Also, it is quite useful to look at some previous elements already implemented in Real-ESSI.

204.6.1 Introduction

This document provides detailed description of steps for adding a new element into the Real-ESSI Simulator. New Element Template source (.ccp) and header (.h) files can be located inside source code in CompGeoMechUCD_FiniteElements directory, and are also shown below in subsection 204.6.3 and subsection 204.6.4.

The list of all the steps to be followed are listed below

1. SubSec ??:: Creating New Element Directory and Linking to Real-ESSI
2. SubSec 204.6.3:: Writing the New Element Header File
3. SubSec 204.6.4:: Writing the New Element Source File
4. SubSec ??:: Setting the ELE_TAG_NewElement class tag and its description
5. SubSec 204.6.6:: Integrating new element with parser.
6. SubSec 204.6.7:: Compiling Real-ESSI
7. SubSec ??:: Verification of Implementation

These steps are shown in each sub-section. The first step starts creating a directory for the new element. After step [3], the new element would be linked with Real-ESSI source code. So, its good to start compiling (step [6]) and fixing bugs rather than going to step [4] or further ahead.

204.6.2 Getting Started:: Creating New Element Directory

The new element can be added in CompGeoMechUCD_FiniteElements directory of Real-ESSI source. A directory for new element lets say NewElement must be created. The next step is to add CMakeLists.txt and place into that directory. The contents of the cmake file is

```
# Builds all the CompGeoMechUCD_FiniteElements/NewElement module
# message("scanning newelements module...")
BUILD_LIB("newelements" ESSI_LIBS)
```
Also, in that directory new element header (NewElement.h) and source files (NewElement.cpp) must be placed. The contents of the cpp files are shown in subsection 204.6.3 and subsection 204.6.4.

The new element then must be included to the header file of Real-ESSI Elements i.e. CompGeoMechUCD_FiniteElements.h. The element header file is loaded in CompGeoMechUCD_FiniteElements directory. So, just add the new element header as

```c
#include "./NewElement/NewElement.h"
```

This would link the new element source code to Real-ESSI. Next is to write the source code and header files of the new element.
204.6.3 Element Header File

The header file is self documented (read fully and carefully).

```cpp
// Rename the header guard
#ifndef NewElementTemplate_h
#define NewElementTemplate_h

#include <Element.h>
#include <Matrix.h>
#include <Vector.h>

class Node;
class Channel;

class NewElementTemplate: public Element
{

public:
    // Constructor
    NewElementTemplate(int tag, int node1, .....); //You must implement this
    // Empty constructor
    NewElementTemplate();
    //Destructor
    ~NewElementTemplate();

    /******************************************************/
    /********* Functions to obtain information about dof & connectivity *******
    /******************************************************/
    // returns the number of external nodes of the element
    int getNumExternalNodes(void) const;
    // returns the ID of external nodes of the element
    const ID &getExternalNodes(void);
    // returns the node pointers array to the nodes of the elements
    Node **getNodePtrs(void);
    // return the total number of degrees of freedom for the element
    int getNumDOF(void);
    // returns the DofList containing number of degrees of freedom for each node
    const ID &getDofList();
    // update all the necessary variables before simulation starts
    void setDomain(Domain *theDomain);
    /******************************************************/
    /********* Functions to set the state of the element **********/
    /*******************************************************/
    // Functions to update the state of the element on obtaining convergence
    int commitState(void);
    // Function to revert to the last committed (converged) state
    int revertToLastCommit(void);
    // Function to revert to the start of the state of the element at the ←
    // beginning of simulation
    int revertToStart(void);
    // Update the element variables for each iteration

};
```
int update(void);
// Remove the load from element
void zeroLoad(void);
// Add Element load
int addLoad(ElementalLoad *theLoad, double loadFactor);
// Send Current Intertial Load of the element
int addInertiaLoadToUnbalance(const Vector &accel);

/** Functions to obtain stiffness, mass and residual **/

const Matrix &getTangentStiff(void);
// return the current tangent stiffness of the element
const Matrix &getInitialStiff(void);
// return the initial tangent stiffness of the element
const Matrix &getDamp(void);
// return the damping stiffness of the element
const Matrix &getMass(void);
// return the mass of the element
const Vector &getResistingForce(void);
// return the resisting force of the element (static case)
const Vector &getResistingForceIncInertia(void);
// return the resisting force of the element (dynamic case)

/** Functions to implement parallel processing **/

int sendSelf(int commitTag, Channel &theChannel);
// Send the variables to the other CPU in a unique order
int receiveSelf(int commitTag, Channel &theChannel, FEM_ObjectBroker &theBroker);

/** Function to Print information about elemnt **/

void Print(ostream &s, int flag = 0);
// Check element correctness
int CheckMesh(ofstream &);
// Give the element a name
std::string getElementName() const
{
    return "NewElementTemplate";
}

/** Generate Output of the Element **/

/* No. of Element Outputs should be as per the Element_Class_tag_Desc */
/* See the classtags.h for more description on encoding of Class_tag */
/* For Optimization all the information about elements are encoded */
/* in the Element_Class_tag Description */
NOTE:- Element_Class_Description [see classTags.h] must be obeyed

// Declare if there is element output except at gauss points
const vector<float> &getElementOutput() ;

// Declare only if there is any gauss point and there is 18 outputs
// per gauss points i.e stress, strain and plastic strain
const vector<float> &getGaussOutput();

// Send the Gauss Coordinates of the Elements
Matrix &getGaussCoordinates(void);

protected:

//Implementation-specific member functions...
// Should be protected, because they're not going to be called
// from outside the class, but you might want to inherit them

private:

// All data must be private. Provide setter and getter methods if
// this class interacts with other classes.

// Declare if there is element output except at gauss points
static vector<float> Element_Output_Vector() ;

// Declare only if there is any gauss point and there are 18
// outputs per gauss points i.e stress, strain and plastic strain
static vector<float> Gauss_Output_Vector();

// Contains information about Number of Dof for each node of the element
static ID DofList;

# endif
204.6.4 Element Source File

The source file is self documented (read fully and carefully).

```cpp
#include <NewElement.h>

// Must define the class tag for
// the new element in this file.
#include <classTags.h>

// NOTE!! Follow the Element_Class_Desc Encoding
// See classTags.h for more details about encoding
// Declare if there is element output except at gauss points
vector<float> NewElementTemplate::Element_Output_Vector(number_of_Element_outputs);
// Declare only if there is any gauss point and there is
// 18 outputs per gauss points i.e stress, strain and plastic strain
vector<float> NewElementTemplate::Gauss_Output_Vector(number_of_gauss_points*18);
// Contains information about Number of Dof for each node of the element
ID NewElementTemplate::DofList(number_of_element_nodes);

//****************************************************************
// Constructor. Receive all input parameters. Should not allocate resources!
// * Input: Defined by user. At least should receive an integer tag, so that base
// class can be initialized.
// * Output: void
NewElementTemplate::NewElementTemplate(int tag, int node1, ....):
    Element(tag, ELE_TAG_NewElement),
    // add more initializers
{
    //ATTENTION!
    // ELE_TAG_NewElement !! Define the class tag in classTags.h
    // with provided encoding formula
    // for setting the material id to the element
    this->setMaterialTag(material->getTag());
    // fill DofList container with number of dofs for each node
    you must implement
}

//****************************************************************
// Empty constructor. Create an empty element (with possibly a bad state)
// * Input: Defined by user. At least should receive an integer tag,
// so that base class can be initialized.
// * Output: void
NewElementTemplate::NewElementTemplate():
    Element(0, ELE_TAG_NewElement),
    // add more initializers setting internal variables to null values
{
    //ATTENTION!
    // ELE_TAG_NewElement !! Define the class tag in classTags.h
    // with provided encoding formula
    // fill DofList container with number of dofs for each node
    you must implement
}
```
48 }
49
50 //**************************************************************************
51 // Destructor. Deallocate resources used by element.
52 // * Input: void
53 // * Output: void
54 NewElementTemplate::~NewElementTemplate()
55 {
56     you must implement
57 }
58
59 //**************************************************************************
60 // returns the number of nodes of the element.
61 // * Input: void
62 // * Output: number of nodes
63 int NewElementTemplate::getNumExternalNodes(void) const
64 {
65     you must implement
66     return number_of_nodes;
67 }
68
69 //**************************************************************************
70 // Return an ID (integer vector) with the external nodes
71 // * Input: void
72 // * Output: ID with tags of external nodes
73 const ID &NewElementTemplate::getExternalNodes(void) const
74 {
75     you must implement
76     return external_nodes;
77 }
78
79 //**************************************************************************
80 // Return pointer array to the nodes
81 // * Input: void
82 // * Output: node pointer array.
83 Node **NewElementTemplate::getNodePtrs(void)
84 {
85     you must implement
86     return nodes;
87 }
88
89 //**************************************************************************
90 //Return the number of dofs in the element.
91 // * Input: void
92 // * Output: number of dofs (sum of dofs over all of element's nodes)
93 int NewElementTemplate::getNumDOF(void)
94 {
95     you must implement
96 }
97
98 //**************************************************************************
//Return the number of dofs in the element.
// * Input: void
// * Output: DofList containing number of degrees of freedom for each node
const ID &getDofList(){
  you must implement
  return this->DofList;
}

//****************************************************************
// Receives a domain pointer, and sets the local domain pointer through
// calling the base class setDomain.
// At this point the domain is defined and set, one can allocate resources
// (get nodal pointers, compute some internal variables like lengths, volumes, etc.).
// Usually we'll set the node pointers here (will be needed for getNodePtrs function).
// Also, we'll check that the given nodes are defined (you get a valid pointer to them) and
// that they have the right number of DOFS (implementation specific)
// * Input: domain pointer (see Domain.h)
// * Output: void
void NewElementTemplate::setDomain(Domain *theDomain){
  // check Domain is not null - invoked when object removed from a domain
  if (theDomain == 0){
    //set node pointers to null
  }
  else{
    //Use the domain to set the node pointers...
    //nodePointers[0] = theDomain->getNode(Nd1);
    //nodePointers[1] = theDomain->getNode(Nd2);
    // Check the pointers...
    // if (nodePointers == 0)
    // { // bad error, usually means node was never
    //   return;
    // }
    // Check the number of DOFs
    // if(nodePointers[0]->getNumberDOF() != MY_NUMBER_OF_DOFS)
    // {
    //   print a tantrum
    //   return;
    // }
    // More checks maybe
    // Set the base class domain pointer
```cpp
this->DomainComponent::setDomain(theDomain);
}

// Additionally one can allocate resources at this point.
you must implement
}

//****************************
// Accept current state of the element and save it. (If applicable)
// I this is a gauss-point based element, one calls commitState on
// the material pointers (Gauss points) owned by this element.
// return 0 if success.
// * Input: void
// * Output: error flag, 0 if success
int NewElementTemplate::commitState(void)
{
    you must implement
    return 0;
}

//****************************
// Revert the state of the element to the last committed state.
// Must call for gausspoints if needed.
// * Input: void
// * Output: error flag, 0 if success
int NewElementTemplate::revertToLastCommit(void)
{
    you must implement
    return 0;
}

//****************************
// Revert the state of the element to the initial state.
// Must call for gausspoints if needed.
// * Input: void
// * Output: error flag, 0 if success
int NewElementTemplate::revertToStart(void)
{
    you must implement
    return 0;
}

//****************************
// Update the state of the element. I.E. compute new tangent stiffness,
// compute new resisting force, advance state variables.
// These changes should not be permanent until commit function is called.
// * Input: void
// * Output: error flag, 0 if success
int NewElementTemplate::update(void)
{
```
```
you must implement
    return 0;
}

//****************************************************************
// (optionl) Set the elemental load to zero.
// * Input: void
// * Output: void
void NewElementTemplate::zeroLoad(void)
{
    // optional to implement
    return 0;
}

//****************************************************************
// (optionl) Add a new elemental load. This will modify the
// resisting force vector.
// * Input: ElementalLoad pointer and a loadFactor.
// * Output: error flag, 0 if success
// Notes:
// * ElementalLoads have a type integer (a tag defined elsewhere) and a Vector ←
// array
// of doubles) with data. Use these to generate the elemental load scaled by the
// load factor (which is also the time-step of the analysis).
int NewElementTemplate::addLoad(ElementalLoad *theLoad, double loadFactor)
{
    // optional to implement
    //
    // Some code to get the type and data. Example is for self_weight.
    // int type;
    // const Vector &data = theLoad->getData(type, loadFactor);
    //
    // if (type == LOAD_TAG_ElementSelfWeight) // Load tags are defined in ←
    // classTags.h
    // {
    // do something, like add a the forces to the resisting_force vector.
    // }
    // return 0;

    // Add intertial terms to resisting force vector.
    // * Input: A vector with nodal accelerations???
    // * Output: error flag, 0 if success
    // Notes: use node pointers to get accelerations from nodes,
    // form an acceleration vector and multiply this with the mass matrix, then
    // add this into the load unbalance (with negative sign, cause it is inertia)
    int NewElementTemplate::addInertiaLoadToUnbalance(const Vector &accel)
    {
        you must implement
        return 0;
    }
Functions to obtain stiffness, mass, damping and residual information

- **Input**: void
- **Output**: reference to tangent stiffness matrix (of size nDOF x nDOF, where nDOF = NewElementTemplate::getNumDOF);
- **Pro tip**: If this matrix computes the tangent stiffness, then it can be stored as a static member variable so that all elements share the same memory space (each element overwrites the tangent). This saves memory.

```cpp
const Matrix &NewElementTemplate::getTangentStiff(void)
{
    you must implement
    return K;
}
```

Functions to obtain initial stiffness

- **Input**: void
- **Output**: reference to initial tangent stiffness matrix (of size nDOF x nDOF, where nDOF = NewElementTemplate::getNumDOF);

```cpp
const Matrix &NewElementTemplate::getInitialStiff(void)
{
    you must implement
    return *K;
    // --suggested variable name for stiffness :).
    // Will be a pointer, so that it can be after construction.
}
```

Optional: If element provides its own damping matrix, then this function returns it

- **Input**: void
- **Output**: reference to damping stiffness matrix (of size nDOF x nDOF, where nDOF = NewElementTemplate::getNumDOF);

```cpp
const Matrix &NewElementTemplate::getDamp(void)
{
    // optional to you must implement
}
```

Optional: If element provides its own damping matrix, then this function returns it

- **Input**: void
- **Output**: reference to damping stiffness matrix (of size nDOF x nDOF, where nDOF = NewElementTemplate::getNumDOF);

```cpp
const Matrix &NewElementTemplate::getMass(void)
{
    // optional to implement
}
```
//****************************************************************
// (optional) If element provides its own damping matrix,
// then this function returns it
// * Input: void
// * Output: reference to damping stiffness matrix (of size nDOF x nDOF,
// where nDOF = NewElementTemplate::getNumDOF);
const Vector &NewElementTemplate::getResistingForce(void)
{
    you must implement
}

//****************************************************************
// (optional) Add inertial terms to resisting force.
// * Input: void
// * Output: Vector of doubles with new resisting force.
// Note: Regularly, this function calls getResistingForce() and then
// adds inertial terms.
const Vector &NewElementTemplate::getResistingForceIncInertia(void)
{
    // (optional to implement)
}

//****************************************************************
// (optional, a must if you want to do parallel processing)
// Send all state data of the element through a channel (usually an MPI_Channel)
// * Input: a reference to the Channel to use.
// * Output: error flag, 0 if success
// Note: This function is usually very involved, and should do a lot of checking
// for pointers and for success of the send.
// Note2: setDomain(...) *might* not be called before using this function.
int NewElementTemplate::sendSelf(int commitTag, Channel &theChannel)
{
    // Useful constructs
    // int error_flag;
    // error_flag = theChannel.sendVector(0, 0, double_data); // the first two ←
    // parameters are deprecated
    //
    // Check that error_flag is not < 0
    //
    // theChannel.sendID(0, 0, integer_data); // the first two parameters are ←
    // deprecated
    //
    // Check that error_flag is not < 0
    you must implement
    return 0;
}

//****************************************************************
// (optional, a must if you want to do parallel processing)
// Receive all state data of the element through a channel
// (usually an MPI_Channel). This data comes from an element
// that is calling sendSelf in some other process.
// * Input: a reference to the Channel to use.
// * Output: error flag, 0 if success
// Note: This function is called after setDomain() so all resources should be made available.
int NewElementTemplate::receiveSelf(int commitTag, Channel &theChannel, FEM_ObjectBroker &theBroker)
{
    // Useful constructs
    // int error_flag;
    // theChannel.receiveVector(0, 0, double_data); // the first two parameters are deprecated
    // Check that error_flag is not < 0
    // theChannel.receiveID(0, 0, integer_data); // the first two parameters are deprecated
    // Check that error_flag is not < 0
    you must implement
    return 0;
}

//****************************************************************
// Print out element info
// * Input: an ostream to print stuff into, and a flag
// * Output: void
// Print stuff into the ostream by using the "<<" operator.
// The flag can be used to request different levels of printing, ie.
// a flag of 0 might be very basic information, while flag > 0 might give increasing amount of info.
void NewElementTemplate::Print(ostream &s, int flag = 0)
{
    you must implement
}

//****************************************************************
// Check element correctness
// * Input: an ostream to print stuff into (print details of what is being checked here.)
// * Output: an error flag (<0 if element is not right in some way)
// Note: be verbose print element tag, etc. Print out only if an error is encountered.
int NewElementTemplate::CheckMesh(ofstream &)
{
    you must implement
}

//****************************************************************
// Output interface functions
// * Input: void
// * Output: Vector (array of doubles) with the element output.
const vector<float> &NewElementTemplate::getElementOutput()
{
    Fill the Element_Output_Vector
    return Element_Output_Vector;
}

const vector<float> &NewElementTemplate::getGaussOutput()
{
    Fill the Gauss_Output_Vector
    NOTE!!! - Exactly 18 outputs should be there per gauss point
    return Gauss_Output_Vector;
}

Matrix &NewElementTemplate::getGaussCoordinates(void)
{
    you must implement
}

// Add you own member functions at the end!
204.6.5 Element Class Tag Description

This subsection describes how to set up the $ELE\_TAG\_NewElement$ for the new element. All the tags for element, material, load etc, must be included in $ClassTags.h$ located in $ModifiedOpenSeesServices$ directory. Each element has unique identifiers:

- **Element Tag ::** It is a unique tag given to each new element type. The element tags are in sequential order. So the new tag must be the next available tag in sequence.

- **Element Tag Description ::** It is an encoding containing information about the elements such as, type of element, number of nodes, number of gauss points, number of outputs etc. The element tag has 9 digits and follows a strict encoding a shown below

```plaintext
// All elements class tags would be in serial
// numbers from 1-N for optimization
/*********************************************************************
* Desc is [Dimension][N. Nodes][Dof per nodes][N. Gauss][No.of Outputs]
* <1-digit> <2-digit> <1-digit> <3-digit> <2-digit>
* - - - - - - - - - - - - - -- - - - - - - - -- - - - -
* [ElementCategory] = <num_of_digits = 1> Category of the element
  1-> Structural Elements
  2-> Contact Elements
  3-> Brick Elements
  4-> Special elements
* [N. Nodes] = <num_of_digits = 2> Number of nodes in elements
* xx-> number of nodes
* [Dof per nodes] = <num_of_digits = 1> Degree of freedom per node
  x-> DOFS per node
* [N. Gauss] = <num_of_digits = 3> Number of gauss points in elements
* xxx-> number of gauss points
* [No.of Outputs] = <num_of_digits = 2> no. of outputs other than at gauss points.
  * xx-> no. of outputs other than at gauss points.
* Default Features
  * 1) Per gauss point there are in total 18 outputs
  * of stress, plastic strain and total strain
  * 2) No.of Outputs -> here means the extra outputs by an element
  * except gauss points. For example:
  * for eight node brick there is 000 No. of Outputs.
*********************************************************************/
```

The above encoding ensures that each element tag is unique and follows a strict format, making it easier to manage and identify different elements in a simulation or analysis.
36 //### NOTE!! :- Every Element have a responsibility to set
37 //### their tag_description Array. Based on the above encoding
38 //### NOTE!! :- Also increase the ELE_TAG_DESC_ARRAY_SIZE to the
39 //### number of element tags
40 //### -----------------------------------------

For example: Eight node brick element has element tag description as 308300800. Simple truss element has 102300002 as element tag description.

In order to set up the element tag, look for the next available element tag. Usually, the next available element tag would be equal to the ELE_TAG_DESC_ARRAY_SIZE which can be found inside ClassTag.h file below the initial element tags. The available number should be added as new element tag and the ELE_TAG_DESC_ARRAY_SIZE should be increased by 1.

This should be followed by appending the element class tag description in ELE_TAG_DESC_ARRAY.

All the steps are shown below,

ClassTags.h Before:

```c
......
#define ELE_TAG_DispBeamColumn3d 94 // 102600012
#define ELE_TAG_Cosserat_8node_brick 95 // 408600800

#define ELE_TAG_DESC_ARRAY_SIZE 96

......................
......................

#define ELE_TAG_DESC_ARRAY int ele_tag_desc_array[] = {
  ELE_TAG_DESC_ENCODING, \\
  100000000, \\
  308300800, \\
  308400800, \\
  308700800, \\
  308300100, \\
  308400100, \\
  .......
  202300009, \\
  102300002, \\
  102600024, \\
  103600006, \\
  104600000, \\
  302300100, \\
  102600012, \\
  102600012, \\
  408600800, \\
}
```
ClassTags.h After:

```c
#define ELE_TAG_DispBeamColumn3d 94 // 102600012
#define ELE_TAG_Cosserat_8node_brick 95 // 408600800
#define ELE_TAG_NewElement 96 // XXXXXXXXX

#define ELE_TAG_DESC_ARRAY_SIZE 97

........................
........................

#define ELE_TAG_DESC_ARRAY_SIZE int ele_tag_desc_array[] = \
{ELE_TAG_DESC_ENCODING, \
 100000000, \ 
 308300800, \ 
 308400800, \ 
 308700800, \ 
 308300100, \ 
 308400100, \ 
 ........................
 202300009, \ 
 102300002, \ 
 102600024, \ 
 103600006, \ 
 104600000, \ 
 302300100, \ 
 102600012, \ 
 102600012, \ 
 408600800, \ 
 XXXXXXXXX, \ 
}
```
204.6.6 Integrating New Finite Element into Parser

Next is to integrate the new element with the parser using `feiparser.yy,feiparser.l` files located in DSL directory located in Real-ESSI source. This step requires some knowledge of yacc and lex. The NewElement DSL should added along with other defined element DSL’s in `feiparser.yy` file. A typical parser for an element looks like as shown below:

```plaintext
1 | TEXTNUMBER exp TYPE NewElement WITH NODES
2 | '(' exp ',' exp ',' exp ',' exp ')' |
3 | USE MATERIAL TEXTNUMBER exp
4 | parameter1 '=' exp
5 | parameter2 '=' exp
6 |
7 | { args.clear(); signature.clear();
8 | args.push_back($2); signature.push_back(this_signature("number", \→
9 | &isAdimensional));
10 | args.push_back($8); signature.push_back(this_signature("node1", \→
11 | &isAdimensional));
12 | args.push_back($10); signature.push_back(this_signature("node2", \→
13 | &isAdimensional));
14 | args.push_back($12); signature.push_back(this_signature("node3", \→
15 | &isAdimensional));
16 | args.push_back($14); signature.push_back(this_signature("node4", \→
17 | &isAdimensional));
18 | args.push_back($19); signature.push_back(this_signature("material", \→
19 | &isAdimensional));
20 |
21 | args.push_back($22); signature.push_back(this_signature("parameter1", \→
22 | &isThisUnit<-1,3,1>));
23 | //L^3*T/M
24 | args.push_back($25); signature.push_back(this_signature("parameter2", \→
25 | &isThisUnit<-1,3,1>));
26 |
27 | $$ = new FeiDslCaller8<int,int,int,int,int,int,double,double>($add_element_new_element, args, signature, \→
28 | "add_element_new_element");
29 |
30 | for(int ii = 1;ii <=8; ii++) nodes.pop();
31 | nodes.push($$);
32 |
33 | }
```

where, the code for DSL of the element corresponds as:

```plaintext
| TEXTNUMBER exp TYPE NewElement WITH NODES
| '(' exp ',' exp ',' exp ',' exp ')' |
| USE MATERIAL TEXTNUMBER exp
| parameter1 '=' exp
| parameter2 '=' exp
```

which would look like the following in DSL language

```plaintext
add element # <.> type NewElement with nodes
(<>), (<.), (<.), (<.>) use material # 1
parameter1 = <.>
parameter2 = <.>
```

In the above DSL each word represents a token which must be included in `feiparse.yy` and defined in `feiparse.l`. There are some already defined tokens, which needs not to be defined. One can find if the token exits by searching it in `feiparser.l` or `feiparse.yy`.

In the above DSL, the tokens are TEXTNUMBER, TYPE, NewElement, WITH, NODES, USE, MATERIAL, TEXTNUMBER, parameter1 and parameter2. Among them, some of them like TEXTNUMBER, TYPE, NODES, .. etc are already defined and could be searched. But the tokens NewElement, parameter1 and parameter2 needs to be defined. First, these all undefined tokens needs to be included in `feiparse.yy` and then defined in `feiparser.l`.

### 204.6.6.1 feiparser.yy

The new tokens must be added in the beginning of the feiparser, where other tokens are defined.

The new tokens can be included as described below.

```plaintext
// Tokens for elements
%token EightNodeBrick TwentyNodeBrick TwentySevenNodeBrick
%token NewElement

// Element options tokens
%token porosity alpha rho_s rho_f k_x k_y k_z K_s K_f pressure cross_section shear_modulus
%token friction_ratio maximum_gap
%token parameter1 parameter2
```

### 204.6.6.2 feiparser.l

The tokens needs to be defined as the following.

```plaintext
"HardContact" {return token::HardContact;}
"SoftContact" {return token::SoftContact;}
"NewElement" {return token::NewElement;}
```

### 204.6.6.3 Argument Stack, Signature and Units

Returning back to the code in parser:
Each variable or parameter in Real-ESSI has units. So, for each of the variables UNIT must be specified. In the above code, \texttt{args} is a stack that should be filled with the tokens that are the parameters of element. The first step is to clear the \texttt{args} and the \texttt{signature} stacks. Pushing the element parameter tokens is done as described below

```cpp
| TEXTNUMBER exp TYPE NewElement WITH NODES
| ( exp ',', exp ',', exp ',', exp ')
| USE MATERIAL TEXTNUMBER exp
| parameter1 '=" exp
| parameter2 '=" exp
| {
|    args.clear(); signature.clear();
|    args.push_back($2); signature.push_back(this_signature("number", ← &isAdimensional));
|    args.push_back($8); signature.push_back(this_signature("node1", ← &isAdimensional));
|    args.push_back($10); signature.push_back(this_signature("node2", ← &isAdimensional));
|    args.push_back($12); signature.push_back(this_signature("node3", ← &isAdimensional));
|    args.push_back($14); signature.push_back(this_signature("node4", ← &isAdimensional));
|    args.push_back($19); signature.push_back(this_signature("material", ← &isAdimensional));
|    args.push_back($22); signature.push_back(this_signature("parameter1", ← &isThisUnit<-1,3,1>));
|    //L^3*T/M
|    args.push_back($25); signature.push_back(this_signature("parameter2", ← &isThisUnit<-1,3,1>));
|    $$ = new FeiDslCaller8<int,int,int,int,int,int,
|       double,double>(&add_element_new_element, args, signature, ← "add_element_new_element");
|    for(int ii = 1; ii <= 8; ii++) nodes.pop();
|    nodes.push($$);
| }

Here, $2$ responds to the the second token i.e exp after \texttt{TEXTNUMBER}. Similarly, $5^{th}$ token in \texttt{TYPE}. In signature, the string can be anything, but should usually be the parameter name. The last
is enforcing the units for each of the parameters by defining the required units. There are many units, such as \textit{isAdimensional}, \textit{isMass}, \textit{isLength}, \textit{isTime}, \textit{isFrequency}, \textit{isArea}, \textit{isVolume}, \textit{isForce}, \textit{isEnergy}, \textit{isTorque}, \textit{isPressure}, \textit{isBodyForce}, \textit{isDensity}, \textit{isVelocity}, \textit{isAcceleration}, \textit{isAreaMomentOfInertia}, \textit{isMassMomentOfInertia}. If the parameter has some other units then the other units can be defined as
\[
isThisUnit< m, l, t >
\]
which refers to the unit $M^mL^lT^t$ in standard units. Here, $M, L, T$ are mass, length and time respectively.

### 204.6.6.4 FeiDslCaller

The next step is to send all the parameters to DSL Header File. The header file contains the code to create a new element inside simulation domain. The code that does this is

```cpp
$ = new FeiDslCaller8<int,int,int,int,int,int,double,double>(
    &add_element_new_element, args,
    signature, "add_element_new_element");
```

For(int ii = 1; ii <= 8; ii++) nodes.pop(); nodes.push($$);

\textit{FeiDslCaller} takes all the arguments and passes to the \textit{add_element_new_element.h} header file. In the above code, the \textbf{number} 8 in function \textit{FeiDslCaller} corresponds to the total number of arguments to the element and following that, the type of the arguments is defined. Here, the type of 8 arguments are \textit{int}, \textit{int}, \textit{int}, \textit{int}, \textit{int}, \textit{int}, \textit{double}, \textit{double}. The last step is the remove everything from the node stack by popping it equal to the number of times of argument and then finally pushing the \textit{FeiDslCaller} to the nodes stack.

### 204.6.6.5 New DSL Header File

The header file of the new DSL must be created in /API/\textit{MODELING} directory. The header file must also be included in /API/api.h header file. Here, the header file \textit{add_element_new_element.h} has been created, which is called by \textit{FeiDslCaller}.

```cpp
int add_element_new_element(int ElementNumber, int Node1, int Node2, int Node3, int Node4, int MaterialId, double Parameter1, double Parameter2)
{
    Element* theElement = 0;
    theElement = new NewElement(ElementNumber, Node1, Node2, Node3, Node4, MaterialId, Parameter1, Parameter2);
    if (theElement == NULL){
```
9     cerr << "Error: (add_element_new_element)
10     memory allocation problem for theElement!" << endl;
11     return -1;
12
13     }
14
15     if (theDomain.addElement(theElement) == false){
16         cerr << "WARNING (add_element_new_element)
17         could not add element to the domain\n"
18         cerr << "Element Number: " << ElementNumber << endl;
19         return -1;
20     }
21
22     return 0;
23 

Header file in /API/api.h file needs to be included:

1     // ##########################
2     // New Element [ABCD Month, Year]
3     // ##########################
4     #include "MODELING/add_element_new_element.h"

204.6.7 Compiling Real-ESSI

It is assumed that the person reading this document is developer and thus should already have the Real-ESSI dependencies. To compile Real-ESSI with the new element, it should be build from the beginning. Assuming that the build directory in build inside Real-ESSI source code, the steps to recompile are

1     cd build
2     rm -r *
3     cmake ..
4     make -j 20

204.6.8 Verification of Implementation

Once, the element is fully integrated with Real-ESSI Simulator, the developer should fully verify the implementation by carrying out verification runs. In addition, the developer should be able to run their examples in sequential and parallel and verify the implementation.
Chapter 205

Input, Domain Specific Language

(In collaboration with Prof. José Abell, Dr. Yuan Feng, and Dr. Hexiang Wang)
205.1 Chapter Summary and Highlights

205.2 Introduction

This chapter presents the domain specific language developed for the Real-ESSI. The language was designed with a primary goal of developing FEA models and interfacing them with various Real-ESSI functionalities. In addition to that, syntax is used to self-document models, provide physical-unit safety, provide common flow control structures, provide modularity to scripting via user functions and “include” files, and provide an interactive environment within which models can be created, validated and verified.

The development of Real-ESSI Domain specific language (DSL) (the Finite Element Interpreter, FEI) is based on LEX (Lesk and Schmidt, 1975) and YACC (Johnson, 1975).

Self-documenting ensures that the resulting model script is readable and understandable with little or no reference to the users manual. This is accomplished by providing a command grammar structure and wording similar to what would be used in a natural language description of the problem.

FEA analysis is unitless, that is, all calculations are carried out without referencing a particular unit system. This leaves the task of unit correctness up to the user of FEA analysis. This represents a recurring source of error in FEA analysis. Physical unit safety is enforced in Real-ESSI by implementing all base variables as physical quantities, that is, all variables have a unit associated with it. The adimensional unit is the base unit for those variables which have no relevant unit (like node numbers). Command calls are sensitive to units. For example, the node creation command call expects the node coordinates to be input with the corresponding units (length in this case). Additionally, the programming/command language naturally supports operation with units like arithmetic operations (quantities with different unit types will not add or subtract but may be multiplied). This approach to FEA with unit awareness provides an additional layer of security to FEA calculations, and forces the user to carefully think about units. This can help catch some common mistakes.

The Real-ESSI language provides modularity through the include directive/command, and user functions. This allows complex analysis cases to be parameterized into modules and functions which can be reused in other models.

Finally, an emphasis is placed on model verification and validation. To this end, Real-ESSI provides an interactive programming environment with all the ESSI syntax available. By using this environment, the user can develop tests to detect errors in the model that are not programming errors. For example, the user can query nodes and elements to see if they are set to appropriate states. Also, several standard tools are provided to check element validity (Jacobian, etc.).

1 user functions have not been implemented yet
The ESSI language provides reduced model development time by providing the aforementioned features along with meaningful error reporting (of syntax and grammatical errors), a help system, command completion and highlighting for several open source and commercial text editors.

Some additional ideas are given by Dmitriev (2004), Stroustrup (2005), Niebler (2005), Mernik et al. (2005), Ward (2003), etc.

### 205.3 Domain Specific Language (DSL), English Language Binding

Overview of the language syntax.

- Each command line has to end with a semicolon ";"
- Comment on a line begins with either "//" or "!" and last until the end of current line.
- Units are required (see more below) for all quantities and variables.
- Include statements allow splitting source into several files
- All variables are double precision (i.e. floats) with a unit attached.
- All standard arithmetic operations are implemented, and are unit sensitive.
- Internally, all units are represented in the base SI units \((m \cdot s \cdot kg)\).
- The syntax ignores extra white spaces, tabulations and newlines. Wherever they appear, they are there for code readability only. (This is why all commands need to end with a semicolon).

#### 205.3.1 Running Real-ESSI

At the command line type "essi", to get to the ESSI prompt and start Real-ESSI in interactive mode.

Command line output

```
The Finite Element Interpreter Endeavor

The Real-ESSI Simulator
Modeling and Simulation of Earthquakes, and Soils, and Structures and their Interaction

Sequential processing mode.
```

A number of useful information about Real-ESSI is printed on the screen. From here, commands can be input manually or a file may be included via the include command which is as follows.

```plaintext
1 include "foobar.fei";
```

to include the file foobar.fei.

A more efficient way to start Real-ESSI and analyze an example is to pass input file name to the command line. Real-ESSI command to execute an input file immediately is done by issuing the following command: `essi -f foobar.fei`. This will execute essi directly on input file `foobar.fei`. After executing the file, the essi interpreter will continue in interactive mode unless the command line flag `-n` or `--no-interactive` is set. A list of command line options is available by calling essi from the command line as `essi -h`.

---

**Command line output**

The Finite Element Interpreter Endeavor

The Real-ESSI Simulator
Modeling and Simulation of Earthquakes, and Soils, and Structures and their Interaction

Sequential processing mode.


**Version Branch** : GLOBAL_RELEASE

**Compile Date** : Jun 13 2018 at 14:36:56
Compile User : jeremic
Compile Sysinfo : sokocalo 4.13.0-43-generic x86_64 GNU/Linux
Runtime User : jeremic
Runtime Sysinfo : sokocalo 4.13.0-43-generic x86_64 GNU/Linux
Time Now : Jun 13 2018 at 16:22:08
Days From Release : 0
PostProcessing Compatible Version: ParaView 5.1.2
PostProcessing Compatible Version: ESSI-pvESSI Date: Feb 15 2018 at 11:00:28. Tag: 58fe430a19

Static startup tips:
* Remember: Every command ends with a semicolon ';'.
* Type 'quit;' or 'exit;' to finish.
* Run 'essi -h' to see available command line options.

The Real-ESSI Simulator
Modeling and Simulation of Earthquakes, and Soils, and Structures and their Interaction

Usage: essi [-cfhnsmbFILENAME]
-c --cpp-output : Output cpp version of the model.
-f --filename [FILENAME] : run ESSI on a FILENAME.
-h --help : Print this message.
-n --no-interactive : Disable interactive mode.
-s --set-variable : Set a variable from the command line.
-d --dry-run : Do not execute ESSI API calls. Just parse.
-m --model-name [NAME] : Set the model name from the command line.
-p --profile-report [FILENAME] : Set the filename for the profiler report (and activate lightweight profiling)

Example to set a variable name from command line:
essi -s a=10,b=20,c=30
Runs ESSI with variables a, b, and c set to 10, 20 and 30 respectively.
At this time, only ESSIunits::unitless variables can be set.

205.3.2 Finishing Real-ESSI Program Run

To properly finish Real-ESSI program run, and save and close all the output files, user has to use final, closure command:

```
bye;
```

Command bye; has to be included at the end of input file script, or at the end of each interactive/interpretative session. Command bye; ensures that Real-ESSI program gracefully exits simulation, and

---

that all the output files are properly saved and closed. Proper finishing of simulation using Real-ESSI Simulator is very much necessary, while the choice of command `bye;` is done as an homage to Professor Knuth and his Literate Programming endeavor (Knuth, 1984), that is driving much of the Real-ESSI DSL development.

There are a number of alternative final commands, for example:

```plaintext
exit;
quit;
zdravo;
vozdra;
dvojka;
voljno;
zaijian;
tschuess;
geia-sou;
```

These additional, alternative final commands all have appropriate meaning in different languages: zdravo ↔ здраво; vozdra ↔ воздра; dvojka ↔ двойка; voljno ↔ волно; zaijian ↔ 再见; tschuess ↔ tschüss; geia-sou ↔ γειά σου.

### 205.3.3 Real-ESSI Variables, Basic Units and Flow Control

Variables are defined using the assignment (\(=\)) operator. For example,

```plaintext
var_x = 7; //Results in the variable x be set to 7 (unitless)
var_y = 3.972e+2; //Scientific notation is available.
```

The language contains a list of reserved keywords. Throughout this documentation, reserved keywords are highlighted in blue or red.

All standard arithmetic operations are available between variables. These operations can be combined arbitrarily and grouped together with parentheses.

```plaintext
var_a = var_x + var_y; // Addition
var_b = var_x - var_y; // Subtraction
var_c = var_x * var_y; // Product
var_d = var_x / var_y; // Quotient
var_e = var_y % var_x; // Modulus (how many times x fits in y)
```

The `print` command can be used to display the current value of a variable.

```plaintext
print var_x;
print var_y;
print var_a;
print var_b;
print var_c;
```
6 print var_d;
7 print var_e;

Command line output

```
var_x = 7 []
var_y = 397.2 []
var_a = 404.2 []
var_b = -390.2 []
var_c = 2780.4 []
var_d = 0.0176234 []
var_e = 5.2 []
```

Here the "unit" (sign) [] means that the quantities are unitless.

The command 'whos' is used to see all the currently defined variables and their values. After a fresh start of essi, needed to clear up all the previously defined variables, command who;’ produces a list of predefined variables:

```
ESSI> whos;

Declared variables:
  * Day = 86400 [s]
  * GPa = 1 [GPa]
  * Hour = 3600 [s]
  * Hz = 1 [Hz]
  * MPa = 1 [MPa]
  * Minute = 60 [s]
  * N = 1 [N]
  * Pa = 1 [Pa]
  * Week = 604800 [s]
  * cm = 1 [cm]
  * feet = 0.3048 [m]
  * ft = 0.3048 [m]
  * g = 9.81 [m*s^-2]
  * inch = 0.0254 [m]
  * kN = 1 [kN]
  * kPa = 1 [kPa]
  * kg = 1 [kg]
  * kip = 4448.22 [N]
  * km = 1 [km]
  * ksi = 6.89476e+06 [Pa]
  * lbf = 4.44822 [N]
  * lbm = 0.453592 [kg]
  * m = 1 [m]
  * mile = 1609.35 [m]
  * mm = 1 [mm]
  * pi = 3.14159 []
  * psi = 6894.76 [Pa]
  * s = 1 [s]
```
Predefined variables shown above have a preceding asterisk to show they are locked variables which cannot be modified. The purpose of these locked variables are to provide names for units. Imperial units are also supported as shown above.

The units for variable are shown between the brackets. Note that unit variables have the same name as their unit, which is not the case for user defined variables. Variables preceded by a star (*) are locked variables which can’t be modified.

For example, the variable 'm' defines 'meter'. So to define a new variable L1 which has meter units we do:

```
L1 = 1*m; // Defines L1 to 1 m.
L2 = 40*mm; // Defines L2 to be 40 millimeters.
```

Even though L2 was created with millimeter units, it is stored in base units.

```
print L2; displays
```

```
Command line output
```

```
L2 = 0.04 [m]
```

As additional examples, let us define few forces:

```
F1 = 10*kN;
F2 = 300*N;
F3 = 4*kg*g;
```

Here g is the predefined acceleration due to gravity.

Arithmetic operations do check (and enforce) for unit consistency. For example, `foo = L1 + F1;` produces an error because units are not compatible. However, `bar = L1 + L2;` is acceptable. On the other hand, multiplication, division and modulus, always work because the result produces a quantity with new units (except when the adimensional quantity is involved).

```
A = L1*L2;
Stress_n = F1 / A;
```

Units for all variables are internally converted to SI units ($kg - m - s$) and stored in that unit system. Variables can be displayed using different units by using the \[\] operator. This does not change the variable, it just displays the value of variable with required unit. For example,

```
print Stress_n; //Print in base SI units.
```
print Stress_n in Pa; //Print in Pascal
print Stress_n in kPa; //Print in kilo Pascal

Command line output

\[
\begin{align*}
\text{Stress}_n &= 250000 \text{ [kg} m^{-1} s^{-2}] \\
\text{Stress}_n &= 250000 \text{ [Pa]} \\
\text{Stress}_n &= 250 \text{ [kPa]}
\end{align*}
\]

The DSL provides functions to test the physical units of variables. For example,
\begin{verbatim}
print isForce(F1);
\end{verbatim}
will print an adimensional, Boolean 1 because \( F1 \) has units of force. While,
\begin{verbatim}
print isPressure(F);
\end{verbatim}
will print an adimensional, Boolean 0. The language also provides comparison of quantities with same units (remember all values are compared in SI Units).
\begin{verbatim}
print F1 > F2;
\end{verbatim}
will print an adimensional, Boolean 1 since \( F1 \) is greater than \( F2 \).

The program flow can be controlled with if and while statements, i.e.:
\begin{verbatim}
if (isForce(F1))
{
    print F1; // This will be executed
}
if (isForce(L1))
{
    print L1; // This will not.
}
\end{verbatim}

Note the necessary semicolon (;) at the closing brace. Unlike C/C++, the braces are always necessary. Closing colon is also always necessary.

The "else" statement is also available:
\begin{verbatim}
if (isForce(L1))
{
    print L1; // This will not execute
}
else
{
    print L2; // This will execute instead
}
\end{verbatim}

While loops are also available:
1 i = 0;
2 while( i < 10)
3 {
4     print i;
5     i = i +1;
6 }

205.3.4 Modeling

This section details ESSI modeling commands. Angle brackets <> are used for quantity or variable placeholder, that is, they indicate where user input goes. Within the angle brackets, the expected unit type is given as well, i.e.. <L> means the command expects an input with a value and a length unit. The symbol <.> represents the adimensional quantity.

In addition to that, the vertical bar | ("OR" sign)) is used to separate two or more keyword options, i.e. [a|b|c] is used indicate keyword options a or b or c. The symbol |...| is used to denote where several long options exist and are explained elsewhere (an example of this is available below in a material model definitions).

All commands require unit consistency. Base units, SI or other can be used as indicated below:

- length, symbol \( L \), units \([m, \text{inch, ft}]\)
- mass, symbol \( M \), units \([\text{kg, lbm}]\),
- time, symbol \( T \), units \([\text{s}]\)

Derived units can also be used:
- angle, symbol rad (radian), unit \([\text{dimensionless, } L/L]\]
- force, symbol N (Newton), units \([N, kN, MN, M \times L/T^2]\),
- stress, symbol Pa (Pascal), units \([Pa, kPa, MPa, N/L^2, M/L/T^2]\)
- strain, symbol (no symbol), units \([L/L]\)
- mass density, symbol (no symbol), units \([M/L^3]\)
- force density, symbol (no symbol), units \([M/L^2/T^2]\)

All models have to be named: model name "model_name_string"; This is important as output files are named based on model name.

Each loading stage has to be named as well. A new loading stage\(^2\) is defined like this:

\(^2\)See more in section 101.3.5 on page 90 in Jeremić et al. (1989-2021).
new loading stage "loading stage name string";
In addition to model name, loading stage name is used for output file name for given loading stage.
205.3.4.1 Modeling, Material Model: Adding a Material Model to the Finite Element Model

Adding constitutive material model to the finite element model/domain is done using command:

```
1 add material # <.> type |...|
2   mass_density = <M/L^3>
3   (more model dependent parameters) ;
```

- Material number # (or alternatively No) is a distinct integer number used to uniquely identify this material.
- Mass density should be defined for each material (even if only static analysis is performed, for example if self weight is to be used as a loading stage).
- Depending on material model, there will be additional material parameters that are defined for each material model/type below:

Starting with version 03-NOV-2015 all elastic-plastic material models require explicit specification of the constitutive integration algorithm. More information on this can be found in 205.3.5.15. Only the material linear_elastic_isotropic_3d_LT ignores this option.

Choices for material_type are listed below.
205.3.4.2 Modeling, Material Model: Linear Elastic Isotropic Material Model

The command is:

```
1  add material # <.> type linear_elastic_isotropic_3d
2      mass_density = <M/L^3>
3      elastic_modulus = <F/L^2>
4      poisson_ratio = <.>;
```

where:

- `mass_density` is the mass density of material \([M/L^3]\)
- `elastic_modulus` is an isotropic modulus of elasticity of a material (units: stress)
- `poisson_ratio` is a Poisson's ratio (dimensionless)

More on this material model can be found in Section 104.6.1 on Page 207 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.3 Modeling, Material Model: Cross Anisotropic Linear Elastic Material Model

The command is:

```plaintext
add material # <.> <material_number>
  type linear_elastic_crossanisotropic
  mass_density = <M/L^3>
  elastic_modulus_horizontal = <F/L^2>
  elastic_modulus_vertical = <F/L^2>
  poisson_ratio_h_v = <.>
  poisson_ratio_h_h = <.>
  shear_modulus_h_v = <F/L^2>;
```

where:

- `mass_density` is the mass density of material \([M/L^3]\)

- `elastic_modulus_horizontal` is an anisotropic modulus of elasticity for horizontal plane of a material \([F/L^2]\)

- `elastic_modulus_vertical` is an anisotropic modulus of elasticity for vertical direction of a material \([F/L^2]\)

- `poisson_ratio_h_v` is a Poisson’s ratio for horizontal-vertical directions [dimensionless]

- `poisson_ratio_h_h` is a Poisson’s ratio for horizontal-horizontal directions [dimensionless]

- `shear_modulus_h_v` is a shear modulus for horizontal-vertical directions \([F/L^2]\)

It is assumed that vertical axes is global \(Z\) axes.

More on this material model can be found in Section 104.6.1 on Page 207 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.4 Modeling, Material Model: von Mises Associated Material Model with Linear Isotropic and/or Kinematic Hardening

Implements von Mises family of constitutive models, with linear kinematic and/or isotropic hardening.

The command is:

```
1 add material # <.> type vonMises
2  mass_density = <M/L^3>
3  elastic_modulus = <F/L^2>
4  poisson_ratio = <.>
5  von_mises_radius = <F/L^2>
6  kinematic_hardening_rate = <F/L^2>
7  isotropic_hardening_rate = <F/L^2> ;
```

where:

- `mass_density` is the mass density of material \([M/L^3]\]
- `elastic_modulus` is the elastic modulus of material \([F/L^2]\]
- `poisson_ratio` is the Poisson's ratio material \([\ ]\)
- `von_mises_radius` is the radius of the deviatoric section of the von Mises yield surface \([F/L^2]\]
- `kinematic_hardening_rate` is the rate of the kinematic hardening \([F/L^2]\)
- `isotropic_hardening_rate` is the rate of the kinematic hardening \([F/L^2]\)

More on this material model can be found in Section 104.6.6 on Page 213 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.5 Modeling, Material Model: von Mises Associated Material Model with Isotropic Hardening and/or Armstrong-Frederic Nonlinear Kinematic Hardening

This command is for von Mises family of constitutive models, with Armstrong-Frederick kinematic and/or isotropic hardening.

The command is:

```plaintext
1 add material # <.> type vonMisesArmstrongFrederick
2    mass_density = <M/L^3>
3    elastic_modulus = <F/L^2>
4    poisson_ratio = <.>
5    von_mises_radius = <.>
6    armstrong_frederick_ha = <F/L^2>
7    armstrong_frederick_cr = <.>
8    isotropic_hardening_rate = <F/L^2> ;
```

where:

- `mass_density` is the mass density of material \([M/L^3]\]
- `elastic_modulus` is the elastic modulus of material \([F/L^2]\]
- `poisson_ratio` is the Poisson’s ratio material \([ \ ]\]
- `von_mises_radius` is the radius of the deviatoric section of the von Mises yield surface \([F/L^2]\]
- `armstrong_frederick_ha` controls rate of the kinematic hardening \([F/L^2]\]
- `armstrong_frederick_cr` controls the saturation limit for kinematic hardening \([\text{Dimensionless}]\]
- `isotropic_hardening_rate` is the rate of the kinematic hardening \([F/L^2]\]

More on this material model can be found in Section 104.6.6 on Page 213 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.6 Modeling, Material Model: Drucker-Prager Associated Material Model with Linear Isotropic and/or Kinematic Hardening

This command is for Drucker-Prager family of constitutive models, with linear kinematic and/or isotropic hardening. This material uses associate plastic flow rule.

The command is:

```
add material # <.> type DruckerPrager
mass_density = <M/L^3>
elastic_modulus = <F/L^2>
poisson_ratio = <.>
druckerprager_k = <F/L^2>
kinematic_hardening_rate = <F/L^2>
isotropic_hardening_rate = <F/L^2>
initial_confining_stress = <F/L^2> ;
```

where:

- `mass_density` is the mass density of material [M/L^3]
- `elastic_modulus` is the elastic modulus of material [F/L^2]
- `poisson_ratio` is the Poisson's ratio material [ ]
- `druckerprager_k` slope of the Drucker-Prager yield surface in p-q_m space (equivalent to M parameter) [F/L^2]
- `kinematic_hardening_rate` is the rate of the kinematic hardening [F/L^2]
- `isotropic_hardening_rate` is the rate of the isotropic hardening [F/L^2]

More on this material model can be found in Section 104.6.7 on Page 220 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.7 Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederick Nonlinear Kinematic Hardening

A Drucker-Prager constitutive model with associative plastic-flow rule, Armstrong-Frederick kinematic hardening, and linear isotropic hardening and linear elastic isotropic elasticity law.

The command is:

```plaintext
add material # <.> type DruckerPragerArmstrongFrederickLE
mass_density = <M/L^3>
elastic_modulus = <F/L^2>
poisson_ratio = <.>
druckerprager_k = <F/L^2>
armstrong_frederick_ha = <F/L^2>
armstrong_frederick_cr = <.>
isotropic_hardening_rate = <F/L^2>
initial_confining_stress = <F/L^2>;
```

where:

- `mass_density` is the mass density of material \([M/L^3]\)
- `elastic_modulus` is the elastic modulus of material \([F/L^2]\)
- `poisson_ratio` is the Poisson’s ratio material \([\ ]\)
- `druckerprager_k` slope of the Drucker-Prager yield surface in \(p-q_m\) space (equivalent to \(M\) parameter) \([F/L^2]\)
- `armstrong_frederick_ha` controls rate of the kinematic hardening \([F/L^2]\)
- `armstrong_frederick_cr` controls the saturation limit for kinematic hardening \([\text{Dimensionless}]\)
- `isotropic_hardening_rate` is the rate of the isotropic hardening \([F/L^2]\)
- `initial_confining_stress` initial confining (mean) pressure \([F/L^2]\)

More on this material model can be found in Section 104.6.7 on Page 220 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.8 Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic
Hardening and/or Armstrong-Frederick Nonlinear Kinematic Hardening and Non-
l inear Duncan-Chang Elasticity

A Drucker-Prager constitutive model with associative plastic-flow rule, Armstrong-Frederick kinematic
hardening, and Duncan-Chang non-linear isotropic elasticity law.

The command is:

```plaintext
add material # <.> type DruckerPragerArmstrongFrederickNE
mass_density = <M/L^3>
DuncanChang_K = <.>
DuncanChang_pa = <F/L^2>
DuncanChang_n = <.>
DuncanChang_sigma3_max = <F/L^2>
DuncanChang_nu = <.>
druckerprager_k = <.>
armstrong_frederick_ha = <F/L^2>
armstrong_frederick_cr = <.>
isotropic_hardening_rate = <F/L^2>
initial_confining_stress = <F/L^2>;
```

where:

- mass_density is the mass density of material [M/L^3]
- DuncanChang_K parameter controlling Young’s modulus [<.>]
- DuncanChang_pa reference pressure [F/L^2]
- DuncanChang_n exponent [<.>]
- DuncanChang_sigma3_max maximum value for $\sigma_3$ ($\sigma_3 < 0$) elastic properties are constant for
greater values of $\sigma_3$ [F/L^2]
- DuncanChang_nu Poisson’s ratio [F/L^2]
- druckerprager_k slope of the Drucker-Prager yield surface in $p$-$q_m$ space (equivalent to $M$
  parameter) [F/L^2]
- armstrong_frederick_ha controls rate of the kinematic hardening [F/L^2]
- armstrong_frederick_cr controls the saturation limit for kinematic hardening [Dimensionless]
- isotropic_hardening_rate is the rate of the isotropic hardening [F/L^2]
- initial_confining_stress initial confining (mean) pressure [F/L^2]
More on this material model can be found in Section 104.6.7 on Page 220 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.9 Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Kinematic Hardening

This command defines Drucker-Prager family of constitutive models, with linear kinematic and/or isotropic hardening. This material uses non-associate plastic flow rule.

The command is:

```plaintext
add material # <.> type DruckerPragerNonAssociateLinearHardening
mass_density = <M/L^3>
elastic_modulus = <F/L^2>
poisson_ratio = <.>
druckerprager_k = <F/L^2>
kinematic_hardening_rate = <F/L^2>
isotropic_hardening_rate = <F/L^2>
initial_confining_stress = <F/L^2>
plastic_flow_xi = <.>
plastic_flow_kd = <.>;
```

where:

- `mass_density` is the mass density of material \([M/L^3]\)
- `elastic_modulus` is the elastic modulus of material \([F/L^2]\)
- `poisson_ratio` is the Poisson's ratio material \([\ ]\)
- `druckerprager_k` slope of the Drucker-Prager yield surface in \(p-q_m\) space (equivalent to \(M\) parameter) \([F/L^2]\)
- `kinematic_hardening_rate` is the linear rate of the kinematic hardening \([F/L^2]\)
- `isotropic_hardening_rate` is the linear rate of the isotropic hardening \([F/L^2]\)
- `initial_confining_stress` initial confining (mean) pressure \([F/L^2]\)
- `plastic_flow_xi` governs the amplitude of plastic volume changes. The higher \(\xi\), the higher the dilatancy. If \(\xi = 0\), the material model will only produce deviatoric plastic strains. [.]
- `plastic_flow_kd` governs the size of the dilatancy surface, a cone in the stress space on which no plastic volume changes occur. \(k_d\) governs the size of this cone: if \(k_d\) is equal to zero, the dilatancy surface shrinks to a line (the hydrostatic axis), so that only dilative soil deformation is possible. [.]

More on this material model can be found in Section 104.6.7 on Page 220 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.10 Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Armstrong-Frederick Nonlinear Kinematic Hardening

This command defines Drucker-Prager family of constitutive models, with nonlinear kinematic and/or linear isotropic hardening. This material uses non-associated plastic flow rule.

The command is:

```
1  add material # <.> type DruckerPragerNonAssociateArmstrongFrederick
2       mass_density = <M/L^3>
3       elastic_modulus = <F/L^2>
4       poisson_ratio = <.>
5       druckerprager_k = <F/L^2>
6       armstrong_frederick_ha = <F/L^2>
7       armstrong_frederick_cr = <.>
8       isotropic_hardening_rate = <F/L^2>
9       initial_confining_stress = <F/L^2>
10      plastic_flow_xi = <.>
11      plastic_flow_kd = <.>;
```

where:

- **mass_density** is the mass density of material \([M/L^3]\)
- **elastic_modulus** is the elastic modulus of material \([F/L^2]\)
- **poisson_ratio** is the Poisson's ratio material \([\ ]\)
- **druckerprager_k** slope of the Drucker-Prager yield surface in \(p-q_m\) space (equivalent to \(M\) parameter) \([F/L^2]\)
- **armstrong_frederick_ha** a kinematic hardening parameter, which governs the initial stiffness after the yield \([F/L^2]\)
- **armstrong_frederick_cr** a kinematic hardening parameter. \(\frac{h_a}{c_r}\) governs the limit of the back-stress [Dimensionless]
- **isotropic_hardening_rate** is the rate of the kinematic hardening \([F/L^2]\)
- **initial_confining_stress** initial confining (mean) pressure \([F/L^2]\)
- **plastic_flow_xi** governs the amplitude of plastic volume changes - the higher \(\xi\), the higher the dilatancy. If \(\xi = 0\), the material model will only produce deviatoric plastic strains. [\]
- **plastic_flow_kd** governs the size of the dilatancy surface, a cone in the stress space on which no plastic volume changes occur. \(k_d\) governs the size of this cone: if \(k_d\) is equal to zero, the
dilatancy surface shrinks to a line (the hydrostatic axis), so that only dilative soil deformation is possible. [1]

More on this material model can be found in Section 104.6.7 on Page 220 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

The physical meanings of $h_a$ and $c_r$ are shown in Figure (205.1) and Figure (205.2).

The physical meanings of $h_a$, $c_r$, $\xi$, and $k_d$ are shown in Figure (205.1) and Figure (205.2).
205.3.4.11 Modeling, Material Model: Hyperbolic Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Armstrong-Frederick Nonlinear Kinematic Hardening

This command defines a hyperbolic Drucker-Prager constitutive model, with nonlinear kinematic and/or linear isotropic hardening. This material uses non-associated plastic flow rule.

The command is:

```plaintext
1  add material # <.> type HyperbolicDruckerPragerNonAssociateArmstrongFrederick
2       mass_density = <M/L^3>
3       elastic_modulus = <F/L^2>
4       poisson_ratio = <.>
5       friction_angle = <.>
6       cohesion = <F/L^2>
7       rounded_distance = <F/L^2>
8       armstrong_frederick_ha = <F/L^2>
9       armstrong_frederick_cr = <.>
10      isotropic_hardening_rate = <F/L^2>
11      initial_confining_stress = <F/L^2>
12      plastic_flow_xi = <.>
13      plastic_flow_kd = <.>;
```

where:

- `mass_density` is the mass density of material \([M/L^3]\)
- `elastic_modulus` is the elastic modulus of material \([F/L^2]\)
- `poisson_ratio` is the Poisson’s ratio for material \([\ ]\)
- `friction_angle` is the initial friction angle of the material. If isotropic hardening is present, friction angle will evolve. [rad]
- `cohesion` is a material constant that defines the cohesion of the material \([F/L^2]\)
- `rounded_distance` is the parameter that controls the shape of the rounded apex of yield surface \([F/L^2]\)
- `armstrong_frederick_ha` a kinematic hardening parameter, that governs the initial stiffness after the yield \([F/L^2]\)
- `armstrong_frederick_cr` a kinematic hardening parameter. It is noted that ratio \(\frac{h_a}{c_r}\) controls the asymptote the back-stress, that can be related to the ultimate shear strength \([\ ]\)
- `isotropic_hardening_rate` is the rate of the isotropic hardening \([F/L^2]\)
• initial_confining_stress initial confining (mean) pressure, a small value just get initial stress out of cone zone \( F/L^2 \)

• plastic_flow_xi governs the amplitude of plastic volume changes - the higher \( \xi \), the higher the dilatancy. If \( \xi = 0 \), the material model will only produce deviatoric plastic strains. [1]

• plastic_flow_kd governs the size of the dilatancy surface, a cone in the stress space on which no plastic volume changes occur. \( k_d \) governs the size of this cone: if \( k_d \) is equal to zero, the dilatancy surface shrinks to a line (the hydrostatic axis), so that only dilative soil deformation is possible. [1]

More on this material model can be found in Section 104.6.8.5 on Page 234 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.12 Modeling, Material Model: Rounded Mohr-Coulomb Associated Linear Isotropic Hardening Material Model

The command is:

```
add material # <.> type roundedMohrCoulomb
  mass_density = <M/L^3>
  elastic_modulus = <F/L^2>
  poisson_ratio = <.>
  RMC_m = <.>
  RMC_qa = <F/L^2>
  RMC_pc = <F/L^2>
  RMC_e = <.>
  RMC_eta0 = <.>
  RMC_Heta = <F/L^2>
  initial_confining_stress = <F/L^2>
```

where

- `mass_density` is the mass density of material \([M/L^3]\]
- `elastic_modulus` is the elastic modulus of material \([F/L^2]\]
- `poisson_ratio` is the Poisson's ratio material \([\ ]\)
- `RMC_m` 0 < \(m\) < 1 parameter of the RMC yield function. Controls roundness of apex in \(p-q\) space. \([\ ]\)
- `RMC_qa` \(q_a\) parameter of the RMC yield function. Controls roundness of apex in \(p-q\) space. \([F/L^2]\)
- `RMC_pc` \(p\) pressure offset \([F/L^2]\)
- `RMC_e` \(e\) parameter controls roundness of the deviatoric cross-section of the yield surface. 0.5 < \(e\) <= 1, \(e = 0.5\) results in a triangular deviatoric section while \(e = 1\) is round. \([\ ]\)
- `RMC_eta0` controls the opening of the yield surface \([\ ]\)
- `RMC_Heta` isotropic (linear) hardening of the yield surface \([F/L^2]\)
- `initial_confining_stress` initial confining (mean) pressure \([F/L^2]\)

More on this material model can be found in Section 104.6.9 on Page 237 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.13 Modeling, Material Model: Cam Clay Material Model

The command is:

```plaintext
add material # <.> type CamClay
mass_density = <M/L^3>
M = <.>
lambda = <.>
kappa = <.>
e0 = <.>
p0 = <|F/L^2>
poisson_ratio = <.>
initial_confining_stress = <|F/L^2>
```

where

- `mass_density` is the mass density of material \([M/L^3]\)]
- `e0` void ratio \(e_0\) at the reference pressure, [dimensionless]
- `M` Cam-Clay slope of the critical state line in stress space, [dimensionless]
- `lambda` Cam-Clay normal consolidation line slope, (unit: dimensionless)
- `kappa` Cam-Clay unload-reload line slope, (unit: dimensionless)
- `poisson_ratio` Constant Poisson-ratio
- `p0` Cam-Clay parameter \(p_0\). Tip of the yield surface in \(q-p\) space. \([F/L^2]\)
- `initial_confining_stress` initial confining (mean) pressure \([F/L^2]\)

More on this material model can be found in Section 104.6.10 on Page 238 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.14 Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model

The command is:

```
1. add material # <.> type vonMisesMultipleYieldSurface
2. mass_density = <M/L^3>
3. elastic_modulus = <F/L^2>
4. poisson_ratio = <.>
5. total_number_of_yield_surface = <.>
6. radiuses_of_yield_surface = <string>
7. radiuses_scale_unit = <F/L^2>
8. hardening_parameters_of_yield_surfaces = <string>
9. hardening_parameters_scale_unit = <F/L^2> ;
```

where

- `mass_density` is the mass density of material [M/L^3]
- `elastic_modulus` is the elastic modulus of the material [F/L^2]
- `poisson_ratio` is the constant Poisson-ratio [dimensionless]
- `total_number_of_yield_surface` is the total number of yield surfaces. [dimensionless]
- `radiuses_of_yield_surface` is the radius list of multiple yield surfaces. This parameter gives the radii of each yield surface from the smallest to the biggest. This parameter should be a string which contains the dimensionless radii. The radii should be separated by a blank space or a comma. [string]
- `radiuses_scale_unit` is the unit of each yield surface. This parameter also provides a method to scale up or scale down the radii of each yield surfaces. [F/L^2]
- `hardening_parameters_of_yield_surfaces` is the hardening parameters corresponding to each yield surface. This parameter should be a string which contains the dimensionless hardening parameters. The hardening parameters should be separated by a blank space or a comma. [string]
- `hardening_parameters_scale_unit` is the unit of the each hardening parameter. This parameter also provides a method to scale up or scale down the hardening parameter of each yield surfaces. [F/L^2]
205.3.4.15 Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model that Matches $G/G_{\text{max}}$ Curves

The command is:

```
1 add material #<.> type vonMisesMultipleYieldSurfaceGoverGmax
2 mass_density = <M/L^3>
3 initial_shear_modulus = <F/L^2>
4 poisson_ratio = <.>
5 total_number_of_shear_modulus = <.>
6 GoverGmax = <string>
7 ShearStrainGamma = <string> ;
```

Command Example is

```
1 add material # 1 type vonMisesMultipleYieldSurfaceGoverGmax
2 mass_density = 0.0*kg/m^3
3 initial_shear_modulus = 3E8 * Pa
4 poisson_ratio = 0.0
5 total_number_of_shear_modulus = 9
6 GoverGmax =
7 "1,0.995,0.966,0.787,0.467,0.320,0.109,0.063"
8 ShearStrainGamma =
9 "0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01";
```

where

- `mass_density` is the mass density of material [M/L^3]
- `initial_shear_modulus` is the initial maximum shear modulus, namely, the Gmax. [F/L^2]
- `poisson_ratio` is the constant Poisson-ratio. [dimensionless]
- `total_number_of_shear_modulus` is the total number of shear modulus, including the initial maximum shear modulus. The total number of yield surface is one less than the total number of shear modulus. Namely, (N+1) areas are divided by N surfaces. [dimensionless]
- `GoverGmax` is the G/Gmax from experiments, including the initial shear modulus. Namely, the first element should be 1.0. Each element is dimensionless. The input should be separated by a blank space or a comma. [string]
- `ShearStrainGamma` is the shear strain $\gamma$ corresponding to the GoverGmax. Note that $\gamma = 2\varepsilon$ when the input is prepared. The first element should be 0.0 corresponding to the initial shear modulus. Each element is dimensionless. The input should be separated by a blank space or a comma. [string]
205.3.4.16 Modeling, Material Model: Drucker-Prager Nonassociated Multi-Yield Surface Material Model

The command is:

```plaintext
add material # <.> type DruckerPragerMultipleYieldSurface
mass_density = <M/L^3>
elastic_modulus = <F/L^2>
poisson_ratio = <.>
initial_confining_stress = <F/L^2>
reference_pressure = <F/L^2>
pressure_exponential_n = <.>
cohesion = <F/L^2>
dilation_angle_eta = <.>
dilation_scale = <.>
total_number_of_yield_surface = <.>
sizes_of_yield_surfaces = <string>
yield_surface_scale_unit = <F/L^2>
hardening_parameters_of_yield_surfaces = <string>
hardening_parameters_scale_unit = <F/L^2>;```

where

- `mass_density` is the mass density of material [M/L^3]
- `elastic_modulus` is the elastic modulus of the material [F/L^2]
- `poisson_ratio` is the constant Poisson-ratio [dimensionless]
- `initial_confining_stress` is the initial confining (mean) pressure [F/L^2]
- `reference_pressure` is the reference pressure for the initial modulus. This parameter is usually 101kPa. [F/L^2]
- `pressure_exponential_n` is the exponential number of the pressure dependent modulus. [dimensionless]
- `cohesion` is the attraction force in the soil. [F/L^2]
- `dilation_angle_eta` controls the dilation and compaction of the material. When the stress ratio is smaller than this parameter, plastic compaction takes place. When the stress ratio is greater than this parameter, the plastic dilation takes place. [dimensionless]
- `dilation_scale` controls the rate of the dilation or compaction in the plastic flow. [dimensionless]
- `total_number_of_yield_surface` is the total number of yield surfaces. [dimensionless]
• `radiuses_of_yield_surface` is the radius list of multiple yield surfaces. This parameter gives the radiiuses of each yield surface from the smallest to the biggest. This parameter should be a string which contains the dimensionless radiiuses. The radiiuses should be separated by a blank space or a comma. [string]

• `radiuses_scale_unit` is the unit of the each yield surface. This parameter also provides a method to scale up or scale down the radiiuses of each yield surfaces. [$F/L^2$]

• `hardening_parameters_of_yield_surfaces` is the hardening parameters corresponding to each yield surface. This parameter should be a string which contains the dimensionless hardening parameters. The hardening parameters should be separated by a blank space or a comma. [string]

• `hardening_parameters_scale_unit` The unit of the each hardening parameter. This parameter also provides a method to scale up or scale down the hardening parameter of each yield surfaces. [$F/L^2$]
205.3.4.17 Modeling, Material Model: Drucker-Prager Nonassociated Material Model that Matches $G/G_{\text{max}}$ Curves

The command is:

```
add material # <.> type DruckerPragerMultipleYieldSurfaceGoverGmax
mass_density = <M/L^3>
initial_shear_modulus = <F/L^2>
poisson_ratio = <.>
initial_confining_stress = <F/L^2>
reference_pressure = <F/L^2>
pressure_exponential_n = <.>
cohesion = <F/L^2>
dilation_angle_eta = <.>
dilation_scale = <.>
total_number_of_shear_modulus = <.>
GoverGmax = <string>
ShearStrainGamma = <string>
```

Command Example is

```
add material # 1 type DruckerPragerMultipleYieldSurfaceGoverGmax
mass_density = 0.0*kg/m^3
initial_shear_modulus = 3E8 * Pa
poisson_ratio = 0.0
initial_confining_stress = 1E5 * Pa
reference_pressure = 1E5 * Pa
pressure_exponential_n = 0.5
cohesion = 0.* Pa
dilation_angle_eta =1.0
dilation_scale = 0.0
total_number_of_shear_modulus = 9
GoverGmax =
"1,0.995,0.966,0.873,0.787,0.467,0.320,0.109,0.063"
ShearStrainGamma =
"0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01";
```

where

- `mass_density` is the mass density of material $[M/L^3]$
- `elastic_modulus` is the elastic modulus of the material $[F/L^2]$
- `poisson_ratio` is the constant Poisson-ratio [dimensionless]
- `initial_confining_stress` is the initial confining (mean) pressure $[F/L^2]$
- `reference_pressure` is the reference pressure for the initial modulus. This parameter is usually 101kPa $[F/L^2]$
• pressure_exponential_n is the exponential number of the pressure dependent modulus. [dimensionless]

• cohesion is the attraction force is the soil. \([F/L^2]\)

• dilation_angle_\eta controls the dilation and compaction of the material. When the stress ratio is smaller than this parameter, plastic compaction takes place. When the stress ratio is greater than this parameter, the plastic dilation takes place. [dimensionless]

• dilation_scale controls the rate of the dilation or compaction in the plastic flow. For this automatic \(G/G_{\text{max}}\) match, the dilation scale has to be zero, which means only deviatoric plastic flow is allowed. If the users want to have volumetric dilation, they can match the \(G/G_{\text{max}}\) manually with the other DruckerPragerMultipleYieldSurface command. [dimensionless]

• total_number_of_shear_modulus is the total number of shear modulus, including the initial maximum shear modulus. The total number of yield surface is one less than the total number of shear modulus. Namely, \((N+1)\) areas are divided by \(N\) surfaces. [dimensionless]

• GoverGmax is the \(G/G_{\text{max}}\) from experiments, including the initial shear modulus. Namely, the first element should be 1.0. Each element is dimensionless. The input should be separated by a blank space or a comma. [string]

• ShearStrainGamma is the shear strain \(\gamma\) corresponding to the GoverGmax. Note that \(\gamma = 2\varepsilon\) when the input is prepared. The first element should be 0.0 corresponding to the initial shear modulus. Each element is dimensionless. The input should be separated by a blank space or a comma. [string]
205.3.4.18 Modeling, Material Model: Rounder Mohr-Coulomb Nonassociated Multi-Yield Surface Material Model

The command is:

```plaintext
add material # <.> type RoundedMohrCoulombMultipleYieldSurface
mass_density = <M/L^3>
elastic_modulus = <F/L^2>
poisson_ratio = <.>
initial_confining_stress = <F/L^2>
reference_pressure = <F/L^2>
pressure_exponential_n = <.>
cohesion = <F/L^2>
RMC_shape_k =<.>
dilation_angle_eta = <.>
dilation_scale = <.>
total_number_of_yield_surface = <.>
sizes_of_yield_surfaces = <string>
yield_surface_scale_unit = <F/L^2>
hardening_parameters_of_yield_surfaces = <string>
hardening_parameters_scale_unit = <F/L^2>;
```

where

- `mass_density` is the mass density of material \( [M/L^3] \)
- `elastic_modulus` is the elastic modulus of the material \( [F/L^2] \)
- `poisson_ratio` is the constant Poisson-ratio [dimensionless]
- `initial_confining_stress` is the initial confining (mean) pressure \( [F/L^2] \)
- `reference_pressure` is the reference pressure for the initial modulus. This parameter is usually 101kPa. \( [F/L^2] \)
- `pressure_exponential_n` is the exponential number of the pressure dependent modulus. [dimensionless]
- `cohesion` is the attraction force is the soil. \( [F/L^2] \)
- `RMC_shape_k` controls the shape of the rounded Mohr-Coulomb yield surface. [dimensionless]
- `dilation_angle_eta` controls the dilation and compaction of the material. When the stress ratio is smaller than this parameter, plastic compaction takes place. When the stress ratio is greater than this parameter, the plastic dilation takes place. [dimensionless]
- `dilation_scale` controls the rate of the dilation or compaction in the plastic flow. [dimensionless]
• **total_number_of_yield_surface** is the total number of yield surfaces. [dimensionless]

• **radiiuses_of_yield_surface** is the radius list of multiple yield surfaces. This parameter gives the radiiuses of each yield surface from the smallest to the biggest. This parameter should be a string which contains the dimensionless radiiuses. The radiiuses should be separated by a blank space or a comma. [string]

• **radiiuses_scale_unit** is the unit of the each yield surface. This parameter also provides a method to scale up or scale down the radiiuses of each yield surfaces. \([F/L^2]\)

• **hardening_parameters_of_yield_surfaces** is the hardening parameters corresponding to each yield surface. This parameter should be a string which contains the dimensionless hardening parameters. The hardening parameters should be separated by a blank space or a comma. [string]

• **hardening_parameters_scale_unit** The unit of the each hardening parameter. This parameter also provides a method to scale up or scale down the hardening parameter of each yield surfaces. \([F/L^2]\)
205.3.4.19 Modeling, Material Model: Tsinghua Liquefaction Material Model

The command is:

```
add material # <.> type TsinghuaLiquefactionModel
  mass_density = <M/L^3>
  poisson_ratio = <.>
  initial_confining_stress = <F/L^2>
  liquefaction_G0 = <.>
  liquefaction_EXPN = <.>
  liquefaction_c_h0 = <.>
  liquefaction_mfc = <.>
  liquefaction_mdc = <.>
  liquefaction_dre1 = <.>
  liquefaction_Dre2 = <.>
  liquefaction_Dir = <.>
  liquefaction_Alpha = <.>
  liquefaction_gamar = <.>
  liquefaction_pa = <.>
  liquefaction_pmin = <.>
```

Command Example is

```
add material # 1 type TsinghuaLiquefactionModel
  mass_density = 0.0*kg/m^3
  poisson_ratio = 0.1
  initial_confining_stress = 1E5 *Pa
  liquefaction_G0 = 800
  liquefaction_EXPN = 0.5
  liquefaction_c_h0 = 1.0
  liquefaction_mfc = 1.2
  liquefaction_mdc = 0.4
  liquefaction_dre1 = 0.5
  liquefaction_Dre2 = 1500
  liquefaction_Dir = 0.1
  liquefaction_Alpha = 0.01
  liquefaction_gamar = 0.01
  liquefaction_pa = 1E5
  liquefaction_pmin = 100 ;
```

where

- `mass_density` is the mass density of material [M/L^3]
- `poisson_ratio` is the constant Poisson ratio [dimensionless]
- `initial_confining_stress` is the initial confining (mean) pressure [F/L^2]
- `liquefaction_G0` is initial modulus scale at the reference pressure. For medium dense soil, G0 is 800. [dimensionless]
• *liquefaction_EXPN* is the exponential number of the pressure dependent modulus. [dimensionless]

• *liquefaction_c_h0* is the plastic modulus coefficient. This parameter should be determined by the G/Gmax curve. When the G/Gmax curve is hyperbolic, h is 1.2. The range of h is 0.7-1.2 [dimensionless].

• *liquefaction_mfc* is the slope of the failure surface in p-q plane. The range of $M_{f,c}$ is 1.4-1.8 [dimensionless].

• *liquefaction_mdc* is the slope of the phase transition surface in p-q plane. The range of $M_{d,c}$ is 0.3-1.0 [dimensionless].

• *liquefaction_dre1* is the accumulation coefficient of the reversible dilatancy. This parameter is usually 0.4 [dimensionless].

• *liquefaction_Dre2* is the release coefficient of the reversible dilatancy. This range of $d_{re,2}$ is 1000-1500 [dimensionless].

• *liquefaction_Dir* is the coefficient of irreversible dilatancy. The parameter $d_{ir}$ controls the initial slope of the irreversible strain development with respect to the number of reversible loadings. Intuitively, when $d_{ir}$ is bigger, the soil becomes liquefaction faster. The parameter $d_{ir}$ can be around 0.2 [dimensionless].

• *liquefaction_Alpha* is the limit of the irreversible strain. Intuitively, $\alpha$ controls the maximum strain after the liquefaction. The parameter $\alpha$ can be around 0.03 [dimensionless].

• *liquefaction_gamar* is the maximum shear strain length in one liquefaction loading. Intuitively, this parameter controls the maximum strain size of one loop. This parameter can be around 0.05 [dimensionless].

• *liquefaction_pa* is the reference pressure. Usually, this parameter is 10000 [dimensionless].

• *liquefaction_pmin* is the minimum pressure in the calculation. If the pressure is smaller than $p_{min}$ during the calculation, the pressure will be set to $p_{min}$. This parameter can be 1. Increasing this parameter can avoid the potential numerical errors on small numbers [dimensionless].
### 205.3.4.20 Modeling, Material Model: SANISand Material Model, version 2004

The command is:

```plaintext
add material # <.> type sanisand2004
  mass_density = <M/L^3>
  e0 = <.>
  sanisand2004_G0 = <.>
  poisson_ratio = <.>
  sanisand2004_Pat = <.>
  sanisand2004_p_cut = <.>
  sanisand2004_Mc = <.>
  sanisand2004_c = <.>
  sanisand2004_lambda_c = <.>
  sanisand2004_xi = <.>
  sanisand2004_ec_ref = <.>
  sanisand2004_m = <.>
  sanisand2004_h0 = <.>
  sanisand2004_ch = <.>
  sanisand2004_nb = <.>
  sanisand2004_A0 = <.>
  sanisand2004.nd = <.>
  sanisand2004_z_max = <.>
  sanisand2004_cz = <.>
  initial_confining_stress = <F/L^2>;
```

where

- **MaterialNumber**: Material tag
- **mass_density** is the mass density of material \( [M/L^3] \)
- **sanisand2004_e0** initial void ratio \( [ ] \)
- **sanisand2004_G0** normalized elastic shear modulus \( [ ] \)
- **poisson_ratio** Poisson's ratio \( [ ] \)
- **sanisand2004_Pat** atmospheric pressure \( [F/L^2] \)
- **sanisand2004_p_cut** pressure cut-off ratio \( [F/L^2] \)
- **sanisand2004_Mc** Critical stress ratio at triaxial compression \( [ ] \)
- **sanisand2004_c** tension-compression strength ratio \( c = M_e/M_c \) \( [ ] \)
- **sanisand2004_lambda_c** parameter for critical state line \( [ ] \)
- **sanisand2004_xi** parameter for critical state line \( [ ] \)
• sanisand2004_ec_ref reference void for critical state line[

• sanisand2004_m opening of the yield surface[

• sanisand2004_h0 bounding surface parameter[

• sanisand2004_ch bounding surface parameter[

• sanisand2004_nb bounding surface parameter[

• sanisand2004_A0 dilatancy parameter[

• sanisand2004_nd dilatancy parameter[

• sanisand2004_z_max maximum \( z \) fabric parameter[

• sanisand2004_cz fabric hardening parameter[

• initial_confining_stress is the initial confining stress \( p = -1/3\sigma_{ii} \) and it is positive in compressions (since there is that \(-\) (minus) sign in front of sum of normal stresses \( \sigma_{ii} \) indicial notation summation convention applies) that are positive in tension [stress].

More on this material model can be found in section 104.6.11 on Page 242 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

**Important note**: This material model should be used together with explicit constitutive algorithms, e.g. Forward_Euler or Forward_Euler_Subincrement. For better result, it is suggested to apply strain increments, or sub-increments, smaller than 1e-4.
205.3.4.21 Modeling, Material Model: SANISand Material Model, version 2008

The command is:

```
add material # <.> type sanisand2008
  mass_density = <M/L^3>
  e0 = <.>
  sanisand2008_G0 = <.>
  sanisand2008_K0 = <.>
  sanisand2008_Pat = <.>
  sanisand2008_k_c = <.>
  sanisand2008_alpha_cc = <.>
  sanisand2008_c = <.>
  sanisand2008_lambda = <.>
  sanisand2008_ec_ref = <.>
  sanisand2008_m = <.>
  sanisand2008_h0 = <.>
  sanisand2008_ch = <.>
  sanisand2008_nb = <.>
  sanisand2008_A0 = <.>
  sanisand2008_nd = <.>
  sanisand2008_p_r = <.>
  sanisand2008_rho_c = <.>
  sanisand2008_theta_c = <.>
  sanisand2008_X = <.>
  sanisand2008_z_max = <.>
  sanisand2008_cz = <.>
  sanisand2008_p0 = <F/L^3>
  sanisand2008_p_in = <F/L^3>
  algorithm = explicit (or) implicit
  number_of_subincrements = <.>
  maximum_number_of_iterations = <.>
  tolerance_1 = <.>
  tolerance_2 = <.>;
```

where

- **MaterialNumber**: Number of the ND material to be used;
- **Algorithm**: Explicit (=0) or Implicit (=1);
- **rho**: density;
- **e0**: initial void ratio at zero strain;
- **G0**: Reference elastic shear modulus [stress];
- **K0**: Reference elastic bulk modulus [stress];
- **sanisand2008_Pat**: atmospheric pressure for critical state line;
- `sanisand2008_k_c`: cut-off factor; for \( p < k_c P_{at} \), use \( p = k_c P_{at} \) for calculation of \( G \); (a default value of \( k_c = 0.01 \) should work fine);

- `sanisand2008_alpha_cc`: critical state stress ratio;

- `sanisand2008_c`: tension-compression strength ratio;

- `sanisand2008_lambda`: parameter for critical state line;

- `sanisand2008_xi`: parameter for critical state line;

- `sanisand2008_ec_ref`: reference void for critical state line, \( e_c = e_{\lambda}(p_c/P_{at})^{x_i} \);

- `sanisand2008_m`: opening of the yield surface;

- `sanisand2008_h0`: bounding surface parameter;

- `sanisand2008_ch`: bounding surface parameter;

- `sanisand2008_nb`: bounding surface parameter;

- `sanisand2008_A0`: dilatancy parameter;

- `sanisand2008_nd`: dilatancy parameter;

- `sanisand2008_p_r`: LCC parameter;

- `sanisand2008_rho_c`: LCC parameter;

- `sanisand2008_theta_c`: LCC parameter;

- `sanisand2008_X`: LCC parameter;

- `sanisand2008_z_max`: fabric parameter;

- `sanisand2008_cz`: fabric parameter;

- `sanisand2008_p0`: yield surface size;

- `sanisand2008_p_in`;

- `number_of_subincrements` number of subincrements in constitutive simulation

- `maximum_number_of_iterations` maximum number of iterations
• tolerance_1 Explicit: tolerance for intersection point (distance between two consecutive points)
  Implicit: yield function tolerance

• tolerance_2 Implicit: residual tolerance

More on this material model can be found in Section 104.6.12 on Page 249 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.22 Modeling, Material Model: Cosserat Linear Elastic Material Model

The command is:

```plaintext
add material # <.> type Cosserat_linear_elastic_isotropic_3d
  mass_density = <M/L^3>
  lambda = <F/L^2>
  mu = <F/L^2>
  chi = <F/L^2>
  pi1 = <F>
  pi2 = <F>
  pi3 = <F>
;  
```

- |MaterialNumber| unique material Number.
- |mass_density| the density of the material.
- |lambda|,|mu|,|chi|,|pi1|,|pi2|,|pi3| are the 6 Cosserat elastic constants (Eringen, 2012).

The relations between elastic constants is as follows Eringen (2012). Note the Young’s modulus and the Poisson’s ratio are different from the classical elasticity:

- Young’s modulus \( E = (2\mu + \chi)(3\lambda + 2\mu + \chi) \).
- Shear modulus \( G = \mu + 1/2\chi \).
- Poisson’s ratio \( \nu = \lambda/(2\lambda + 2\mu + \chi) \).
- Characteristic length for torsion \( l_t = ((\pi_2 + \pi_3)/(2\mu + \chi))^{1/2} \).
- Characteristic length for bending \( l_b = (\pi_3/2(2\mu + \chi))^{1/2} \).
- Coupling number \( N = (\chi/2(\mu + \chi)) \)
- Polar ratio \( \Phi = (\pi_2 + \pi_3)/(\pi_1 + \pi_2 + \pi_3) \)

According to Eringen Eringen (2012), the 6 elastic constants should satisfy the following conditions

\[
\begin{align*}
3\lambda + 2\mu + \chi & \geq 0, \quad 2\mu + \chi \geq 0, \quad \chi \geq 0, \\
3\pi_1 + \pi_2 + \pi_3 & \geq 0, \quad \pi_3 + \pi_2 \geq 0, \quad \pi_3 - \pi_2 \geq 0.
\end{align*}
\]
205.3.4.23 Modeling, Material Model: von Mises Cosserat Material Model

The command is:

```
add material # <.> type Cosserat_von_Mises
  mass_density = <M/L^3>
  lambda = <F/L^2>
  mu = <F/L^2>
  chi = <F/L^2>
  pi1 = <F>
  pi2 = <F>
  pi3 = <F>
  plastic_internal_length = <L>
  von_mises_radius = <F/L^2>
  isotropic_hardening_rate = <F/L^2>
```

- **|MaterialNumber|** unique material Number.
- **|mass_density|** the density of the material.
- **|lambda|,|mu|,|chi|,|pi1|,|pi2|,|pi3|** are the 6 Cosserat elastic constants\cite{Eringen2012}.
- **|plastic_internal_length|** is the characteristic length in the plasticity.
- **|von_mises_radius|** is radius of the unified yield surface of force-stress and couple-stress.
- **|isotropic_hardening_rate|** is the rate of isotropic hardening.
205.3.4.24 Modeling, Material Model: Uniaxial Linear Elastic, Fiber Material Model

The command is:

```plaintext
add material # <.> type uniaxial_elastic
   elastic_modulus = <F/L^2>
   viscoelastic_modulus = <mass / length / time> ;
```

where

- |MaterialNumber| unique material number.
- |elastic_modulus| elastic modulus of the material.
- |viscoelastic_modulus| damping tangent.

More on this material model can be found in Section ?? on Page ?? in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

As the name implies, uniaxial_elastic material model works with uniaxial element only. For 3D elements, for example solid brick elements, please use 3D material models, for example, linear_elastic_isotropic_3d.
205.3.4.25 Modeling, Material Model: Stochastic Uniaxial Linear Elastic Model

The command is:

```
add material # <.> type stochastic_uniaxial_elastic uncertain_elastic_modulus = ←
  random variable # <.> elastic_modulus_scale_unit = <F/L^2>;
```

where

- `|uncertain_elastic_modulus|` specify uncertain elastic modulus of the material through a defined random variable.

- `|elastic_modulus_scale_unit|` specify the unit scale factor that would be multiplied with the polynomial chaos coefficients of the random variable.

As the name implies, `stochastic_uniaxial_elastic` material model works with stochastic uniaxial element only.

For example:

```
add material # 1 type stochastic_uniaxial_elastic uncertain_elastic_modulus = ←
  random variable # 1 elastic_modulus_scale_unit = 1*Pa;
```

Add material #1 as `stochastic_uniaxial_elastic` material with uncertain elastic modulus characterized by the polynomial chaos coefficients of random variable 1 and scale factor $1 \times Pa$. 
205.3.4.26 Modeling, Material Model: Stochastic Uniaxial Nonlinear Armstrong Frederick Model

The command is:

```plaintext
1 add material # <.> type stochastic_uniaxial_Armstrong_Frederick
2 constitutive triple product # <.>
3 armstrong_frederick_ha = random variable # <.>
4 armstrong_frederick_ha_scale_unit = <F/L^2>
5 armstrong_frederick_cr = random variable # <.>
```

or

```plaintext
1 add material # <.> type stochastic_uniaxial_Armstrong_Frederick
2 constitutive triple product # <.>
3 armstrong_frederick_ha = random variable # <.>
4 armstrong_frederick_ha_scale_unit = <F/L^2>
5 armstrong_frederick_cr = random variable # <.>
6 polynomial_chaos_terms_ha = <.>
7 polynomial_chaos_terms_cr = <.>
8 polynomial_chaos_terms_incremental_strain = <.>;
```

Note that the difference between these two commands is that the first command would by default use the full polynomial chaos (PC) bases defined in the provided constitutive triple product for probabilistic constitutive modeling. The second command would support user-specified number of polynomial chaos terms for uncertain armstrong_frederick_ha, armstrong_frederick_cr and incremental_strain. This enables users to perform truncation of PC bases for probabilistic constitutive modeling.

The command input parameters are:

- **constitutive triple product #** specifies the ID of the triple product, that would be used in probabilistic constitutive updating. In stochastic finite element method (FEM), the first and second PC basis for this triple product should come from the joint PC representation of uncertain parameters armstrong_frederick_ha and armstrong_frederick_cr. The third PC basis for this triple product should come from the PC representation of uncertain FEM system response, e.g., uncertain structural displacement.

- **armstrong_frederick_ha = random variable #** specifies the uncertain Armstrong Frederick parameter \( ha \) through a defined random variable.

- **armstrong_frederick_ha_scale_unit** specifies the unit scale factor that would be multiplied with the polynomial chaos coefficients of the random variable of uncertain Armstrong Frederick parameter \( ha \).
• armstrong_frederick_cr = random variable # specifies the uncertain Armstrong Frederick parameter cr through a defined random variable.

• polynomial_chaos_terms_ha specifies the number of polynomial chaos basis of uncertain ha involved in the probabilistic constitutive updating.

• polynomial_chaos_terms_cr specifies the number of polynomial chaos basis of uncertain cr involved in the probabilistic constitutive updating.

• polynomial_chaos_terms_incremental_strain specifies the number of polynomial chaos basis of uncertain incremental strain dϵ involved in the probabilistic constitutive updating.

As the name implies, stochastic_uniaxial_Armstrong_Frederick material model works with stochastic uniaxial element only.

For example:

```plaintext
1 add material # 1 type stochastic_uniaxial_Armstrong_Frederick
2 constitutive triple product # 1
3 armstrong_frederick_ha = random variable # 1
4 armstrong_frederick_ha_scale_unit = 1*Pa
5 armstrong_frederick_cr = random variable # 2
6 polynomial_chaos_terms_ha = 10
7 polynomial_chaos_terms_cr = 10
8 polynomial_chaos_terms_incremental_strain = 30;
```

Add material # 1 as stochastic_uniaxial_Armstrong_Frederick material with triple product # 1 for probabilistic constitutive updating.

Uncertain parameter ha is characterized by random variable # 1 using scale unit 1*Pa.

Uncertain parameter cr is characterized by random variable # 2. The number of polynomial chaos basis for uncertain parameters ha, cr and incremental strain in probabilistic constitutive updating are 10, 10 and 30, respectively.
205.3.4.27 Modeling, Material Model: Uniaxial Nonlinear Concrete, Fiber Material Model, version 02

The command is:

```
add material # <.> type uniaxial_concrete02
  compressive_strength = <F/L^2>
  strain_at_compressive_strength = <.>
  crushing_strength = <F/L^2>
  strain_at_crushing_strength = <.>
  lambda = <.>
  tensile_strength = <F/L^2>
  tension_softening_stiffness = <F/L^2>;
```

- |compressive_strength| compressive strength.
- |strain_at_compressive_strength| strain at compressive strength.
- |crushing_strength| crushing strength.
- |strain_at_crushing_strength| strain at crushing strength.
- |lambda| ratio between unloading slope at epscu and initial slope.
- |tensile_strength| tensile strength.
- |tension_softening_stiffness| tension softening stiffness (absolute value) (slope of the tension softening branch).

More on this material model can be found in Section ?? on Page ?? in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.28 Modeling, Material Model: Faria-Oliver-Cervera Concrete Material

The command is:

```plaintext
1 add material No (or #) <material_number>
2   type FariaOliverCerveraConcrete
3   elastic_modulus = <F/L^2>
4   poisson_ratio = <.>
5   tensile_yield_strength = <F/L^2>
6   compressive_yield_strength = <F/L^2>
7   plastic_deformation_rate = <.>
8   damage_parameter_Ap = <.>
9   damage_parameter_An = <.>
10  damage_parameter_Bn = <.>
```

where

- No (or #)<material_number> is a unique material integer number (does not have to be sequential, any unique positive integer number can be used).
- type FariaOliverCerveraConcrete is the material type.
- elastic_modulus is the elastic modulus of material \([F/L^2]\)
- poisson_ratio is the Poisson’s ratio material.
- tensile_yield_strength is the tensile yield strength \([F/L^2]\)
- compressive_yield_strength is the compressive yield strength \([F/L^2]\)
205.3.4.29  Modeling, Material Model: Plane Stress Layered Material

The command is:

```plaintext
add material No (or #) <element_number>
  type PlaneStressLayeredMaterial
  number_of_layers = <.>
  thickness_array = <string>
  thickness_scale_unit = <L>
  with material # <string>
; 
```

where

- No (or #)<material_number> is a unique material integer number (does not have to be sequential, any unique positive integer number can be used).
- type PlaneStressLayeredMaterial is the material type.
- number_of_layers is the number of layers in this layered material.
- thickness_array is the thickness ratio of each individual material.
- thickness_scale_unit set the length unit and the scale factor for the thickness of the layered material.
- material # <string> is the string of predefined individual material tags.
205.3.4.30 Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 01

The command is:

```
1 add material # <.> type uniaxial_steel01
2   yield_strength = <F/L^2>
3   elastic_modulus = <F/L^2>
4   strain_hardening_ratio = <.>
5   a1 = <.>
6   a2 = <.>
7   a3 = <>
8   a4 = <.> ;
```

- `|yield_strength|` yield strength.
- `|elastic_modulus|` initial elastic tangent.
- `|strain_hardening_ratio|` strain-hardening ratio (ratio between post-yield tangent and initial elastic tangent).
- `|a1, a2, a3, a4— isotropic hardening parameters`
  - `|a1|` isotropic hardening parameter, increase of compression yield envelope as proportion of yield strength after a plastic strain of a2*(fy/Ep). ;
  - `|a2|` isotropic hardening parameter (see explanation under a1) ;
  - `|a3|` isotropic hardening parameter, increase of tension yield envelope as proportion of yield strength after a plastic strain of a4*(fy/Ep) ;
  - `|a4|` isotropic hardening parameter (see explanation under a3) ;

More on this material model can be found in Section ?? on Page ?? in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.31 Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 02

The command is:

```plaintext
add material # <.> type uniaxial_steel02
  yield_strength = <F/L^2>
  elastic_modulus = <F/L^2>
  strain_hardening_ratio = <.>
  R0 = <.>
  cR1 = <.>
  cR2 = <.>
  a1 = <.>
  a2 = <.>
  a3 = <>
  a4 = <.> ;
```

- **yield_strength**: yield strength;
- **elastic_modulus**: initial elastic tangent;
- **strain_hardening_ratio**: strain-hardening ratio (ratio between post-yield tangent and initial elastic tangent);
- **R0, cR1, cR2**: control the transition from elastic to plastic branches. Recommended values: R0=between 10 and 20, cR1=0.925, cR2=0.15;
- **a1, a2, a3, a4**: isotropic hardening parameters;
  - **a1**: isotropic hardening parameter, increase of compression yield envelope as proportion of yield strength after a plastic strain of a2*(Fy/E).
  - **a2**: isotropic hardening parameter (see explanation under a1);
  - **a3**: isotropic hardening parameter, increase of tension yield envelope as proportion of yield strength after a plastic strain of a4*(Fy/E).
  - **a4**: isotropic hardening parameter (see explanation under a3);

More on this material model can be found in Section ?? on Page ?? in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.32 Modeling, Material Model: Plane Stress Plastic Damage Concrete Material

This is a plane stress version of the plastic damage concrete model developed by Faria et al. (1998). This material was implemented as part of the endeavor to model reinforced concrete shells, plates and shear walls. It should only be used together with Inelastic Layered Shell Section and 4 Node Shell NLDKGQ/Xin-Zheng-Lu, see page 878.

The command is:

```
add material No (or #) <material_number> type PlasticDamageConcretePlaneStress
  elastic_modulus = <F/L^2>
  poisson_ratio = <.>
  tensile_yield_strength = <F/L^2>
  compressive_yield_strength = <F/L^2>
  plastic_deformation_rate = <.>
  damage_parameter_Ap = <.>
  damage_parameter_An = <.>
  damage_parameter_Bn = <.>
```

where

- No (or #)<material_number> is a unique material integer number (does not have to be sequential, any unique positive integer number can be used).
- elastic_modulus is the elastic modulus of material \([F/L^2]\)
- poisson_ratio is the Poisson’s ratio material.
- tensile_yield_strength is the tensile yield strength \([F/L^2]\)
- compressive_yield_strength is the compressive yield strength \([F/L^2]\)
- plastic_deformation_rate governs the post-yield hardening modulus in the effective (undamaged) space and the plastic strain rate
- damage_parameter_Ap governs the tensile fracture energy and affects the ductility of the tensile response
- damage_parameter_An governs the softening behavior of concrete in compression, it changes the ductility but does not alter the peak strength
- damage_parameter_Bn governs the softening behavior of concrete in compression, it changes both the ductility and the peak strength
205.3.4.33 Modeling, Material Model: Plane Stress Rebar Material

This is a plane stress version of the Uniaxial Nonlinear Steel material. This material was implemented as part of the endeavor to model reinforced concrete shells, plates and shear walls. This model should be used together with Inelastic Layered Shell Section and 4 Node Shell NLDKGV/Xin-Zheng-Lu, see page 878.

The command is:

```
1 add material No (or #) <material_number> type PlaneStressRebarMaterial
2   with uniaxial_material # <.>
3   angle = <degree> ;
```

where

- No (or #)<material_number> is a unique material integer number (does not have to be sequential, any unique positive integer number can be used).
- with uniaxial_material # is the material tag of predefined uniaxial steel material
- angle is the angle of uniaxial steel rebars. The angle is 0 along the direction formed by the first two nodes of a 4 Node Shell element.
205.3.4.34 Modeling, Nodes: Adding Nodes

Nodes can be added to the finite element model.

The command is:

```
1        add node # <.> at (<L>,<L>,<L>) with <.> dofs;
```

For example:

```
1        add node No 1 at (1.0*m, 2.5*m, 3.33*m) with 3 dofs;
```

adds a node number 1 at coordinates \( x = 1.0m, y = 2.5m \) and \( z = 3.33m \) with 3 dofs. The nodes can be of 3dofs \( [u_x, u_y, u_z] \), 4dofs \( [u_x, u_y, u_z, p]\) (u-p elements), 6dofs \( [u_x, u_y, u_z, r_x, r_y, r_z] \) (beams and shells) and 7 dofs \( [u_x, u_y, u_z, p, U_x, U_y, U_z] \) (upU element) types. Description of output for nodes of different dof types can be found in section 206.6.
205.3.4.35 Modeling, Nodes: Adding Stochastic Nodes

Nodes can be added to the stochastic finite element model. Different from deterministic finite element analysis, nodes in stochastic FEM should specify the number of polynomial chaos (PC) terms for each physical nodal degree of freedom (dof).

The command is:

```
1 add node # <.> at (<L>,<L>,<L>) with <.> dofs polynomial_chaos_terms = <.>;
```

Where:

- `polynomial_chaos_terms` specifies the number of polynomial chaos terms for each physical nodal dof.

The stochastic nodes can also be added as:

```
1 add node # <.> at (<L>,<L>,<L>) with <.> dofs polynomial_chaos_terms as random ← field # <.>;
```

Which specifies the number of polynomial chaos terms for each physical nodal dof using the number of Hermite PC basis of a defined random field.

For example:

```
1 add node # 1 at (1.0*m, 0.0*m, 0.0*m) with 3 dofs polynomial_chaos_terms = 10;
```

Add a node # 1 at coordinates $x = 1.0m$, $y = 0.0m$ and $z = 0.0m$ with 3 physical dofs. For each physical dof, the number of terms for polynomial chaos expansion is 10.

```
1 add node # 1 at (1.0*m, 0.0*m, 0.0*m) with 3 dofs polynomial_chaos_terms as ← random field # 2;
```

Add a node # 1 at coordinates $x = 1.0m$, $y = 0.0m$ and $z = 0.0m$ with 3 physical dofs. For each physical dof, the number of terms for polynomial chaos expansion is equal to the number of PC basis of random field # 2.
205.3.4.36 Modeling, Nodes: Define Nodal Physical Group

Physical Group for nodes can be defined as well.

The command is:

```plaintext
1 define physical_node_group "string";
```

For example:

```plaintext
1 define physical_node_group "my_new_node_group";
```

this would create a new physical_node_group with name "my_new_node_group".

Description of output for physical groups can be found in section 206.5.5
205.3.4.37 Modeling, Nodes: Adding Nodes to Nodal Physical Group

Already created nodes can be added to the (any) physical_node_group.

The command is:

```
add nodes (<.>,<.>,...) to physical_node_group "string";
```

For example:

```
add nodes (1,2,3) to physical_node_group "my_new_node_group";
```

this would add node tag (1,2 and 3) to already created physical_node_group "my_new_node_group". Please note that the nodes (1,2 and 3) must be added to the model before they are added to the physical_node_group.

Description of output for physical groups can be found in section 206.5.5.
205.3.4.38 Modeling, Nodes: Removing Nodal Physical Group

Already defined node physical group `physical_node_group` can be removed.

The command is

```plaintext
remove physical_node_group "string";
```

For example:

```plaintext
remove physical_node_group "my_new_node_group";
```

this would delete the `physical_node_group"my_new_node_group"`. 
205.3.4.39 Modeling, Nodes: Print Nodal Physical Group

Printing already defined nodal physical group op physical_node_group is possible too.

The command is:

```
print physical_node_group "string";
```

For example:

```
print physical_node_group "my_new_node_group";
```

this would print the information about physical_node_group "my_new_node_group".

```
PHYSICAL_NODE_GROUP my_new_node_group
[1 2 3]
```
205.3.4.40  Modeling, Nodes: Removing Nodes

Nodes can be removed from the finite element model, for example during excavation, removal of finite elements.

The command is:

```
remove node No (or #) <.>
```

For example:

```
remove node # 1;
```
205.3.41 Modeling, Nodes: Adding Nodal Mass, for 3DOFs and/or 6DOFs

Nodal mass can be added to nodes with 3 DOFs and/or 6DOFs. This is in addition to nodal mass that is obtained from finite elements.

The command for 3DOFs nodes (truss, solids, wall) is:

```
1 add mass to node # <.>
2 mx = <M>
3 my = <M>
4 mz = <M>;
```

Similarly, the command for 6DOFs nodes (beams and shells) is:

```
1 add mass to node # <.>
2 mx = <M>
3 my = <M>
4 mz = <M>
5 Imx = <M*L^2>
6 Imy = <M*L^2>
7 Imz = <M*L^2>;
```
205.3.4.42 Modeling, Finite Element: Adding Finite Elements

The basic structure for adding any finite element is:

```
1 add element No (or #)
2   type <finite_element_type>
3   with nodes (<.), ..., (<.>)| 
4   {element dependent parameters};
```

Choices for `finite_element_type` are listed below
205.3.43 Modeling, Finite Element: Define Finite Element Physical Group

Physical group for finite elements can be defined.

The command is:

```
1 define physical_element_group "string";
```

For example:

```
1 define physical_element_group "my_new_element_group";
```

this would create a new physical_element_group with name "my_new_element_group".

Description of output for physical groups can be found in Section 206.5.5.
205.3.4.44 Modeling, Finite Element: Adding Elements to Physical Element Group

Finite elements, that already exist in the finite element domain, can be added to the physical_element_group.

The command is:

```
add elements (<.>,<.>,...) to physical_node_group "string";
```

For example:

```
add elements (1,2,3) to physical_node_group "my_new_node_group";
```

this would add elements with tags/numbers (1,2 and 3) to already created physical_element_group "my_new_element_group". Please note that the elements (1,2 and 3) must be added to the model before they are added to the physical_element_group.

Description of output for physical groups can be found in Section 206.5.5.
205.3.4.45  Modeling, Finite Element: Remove Physical Finite Element Group

Finite elements can also be removed from the physical_element_group.

The command is:

```
remove physical_element_group "string";
```

For example:

```
remove physical_element_group "my_new_element_group";
```

this would delete the physical_element_group "my_new_element_group".
205.3.4.46 Modeling, Finite Element: Print Physical Finite Element Group

Details of the physical_element_group can be printed.

The commands is:

```
print physical_element_group "string";
```

For example:

```
print physical_element_group "my_new_element_group";
```

this would print the information about physical_element_group "my_new_element_group".

```
PHYSICAL_ELEMENT_GROUP my_new_element_group
[1 2 3]
```
205.3.4.47 Modeling, Finite Element: Remove Finite Element

Finite elements can be removed, for example if modeling requires excavation, removal of finite elements and nodes.

The command is:

```
1 remove element # <.>;
```

For example,

```
1 remove element # 1;
```
205.3.4.48 Modeling, Finite Element: Truss Element

The command is:

```
add element No (or #) <element_number> type truss
    with nodes (n1, n2)
    use material No (or #) <material_number>
    section_area <section_area> [unit];
    mass_density <mass_density> [unit];
```

where

- `No (or #)<element_number>` is a unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- `type truss` is the element type
- `with nodes (n1, n2)` are the 2 nodes (node numbers) defining this element
- `use material No (or #) <material_number>` is the material number which makes up the element. Material has to be a uniaxial material, and it can be either elastic or one of the elastic-plastic materials defined for uniaxial behavior.
- `section_area` is the cross section area \([L^2]\)

Description of output by this element can be found in Section 206.8.1. More on this finite element can be found in Section 102.6 on Page 120 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.49 Modeling, Finite Element: Kelvin-Voigt Truss Element

The command is:

```plaintext
1  add element # <.> type Kelvin_Voigt_Truss
2     with nodes (<.>, <.>)
3        axial_stiffness = <F/L>
4        axial_viscous_damping = <F/L*T>;
```

where make

- No (or #)<element_number> is a unique element integer number, that does not have to be sequential, any unique positive integer number can be used
- type Kelvin_Voigt_Truss is the element type
- with nodes (n1, n2) are the 2 nodes (node numbers) defining this element
- axial_stiffness represents the stiffness in the axial direction, \([F/L]\)
- axial_viscous_damping represents the viscosity, or viscous damping coefficient, in the axial direction, \([F/L \times T]\)

**Note:** Nodes defining this element cannot be at the same location, that is, this is a truss element and direction of this element is calculated from two distinct locations/coordinates of node.
205.3.4.50 Modeling, Finite Element: Shear Beam Element

The command is:

```
add element # <.> type ShearBeam
with nodes (<.>, <.>)
cross_section = <l^2>
use material # <.>;
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- with nodes (n1, n2) are the 2 nodes (node numbers) defining this element.
- use material No (or #) is the material (LT-based material) number which makes up the element.
- section_area is the cross section area [L^2]

Description of output by this element can be found in Section 206.8.3. more on this finite element can be found in Section 102.9 on page 129 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.51 Modeling, Finite Element: Stochastic Shear Beam Element

Add stochastic shear beam element for stochastic finite element analysis.

```plaintext
1 add element # <.> type stochastic_shear_beam with nodes (<.>, <.>)
2   use material # <.>
3   triple product # <.>
4   cross_section = <L^2>
5   mass_density = <M/L^3>;
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used).
- with nodes (n1, n2) are the 2 nodes (node numbers) defining this element.
- use material No (or #) is the stochastic uniaxial material number that makes up the element.
- triple product # specifies the ID of the triple product, that would be used in the formation of elemental stochastic stiffness matrix. In stochastic finite element method (FEM), the first PC basis for this triple product should come from the PC representation of uncertain element stiffness. The second and third PC basis for this triple product should come from the PC representation of uncertain FEM system response, e.g., uncertain structural displacement.
- section_area is the cross section area \([L^2]\).
- mass_density is the density \([M/L^3]\).

For example:

```plaintext
1 add element # 1 type stochastic_shear_beam with nodes (1, 2) use material # 1 ←
   triple product # 1 cross_section = 1*m^2 mass_density = 2000*kg/m^3;
```

Add a stochastic shear beam element # 1 with stochastic nodes 1 and 2 using stochastic uniaxial material # 1.

The cross section of the element is \(1 \text{ m}^2\) and mass density is \(2000 \text{ kg/m}^3\).
### 205.3.4.52 Modeling, Finite Element: Elastic Beam–Column Element

The command is:

```plaintext
add element # <.> type beam_elastic with nodes (<.>, <.>)
  cross_section = <L^2>
  elastic_modulus = <F/L^2>
  shear_modulus = <F/L^2>
  torsion_Jx = <length^4>
  bending_Iy = <length^4>
  bending_Iz = <length^4>
  mass_density = <M/L^3>
  xz_plane_vector = (<.>, <.>, <.> )
  joint_1_offset = (<L>, <L>, <L> )
  joint_2_offset = (<L>, <L>, <L> );
```

![Beam Element, sketch of main geometric components.](image)

Figure 205.3: Beam Element, sketch of main geometric components.

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- type beam_elastic is the element type
• with nodes (n1, n2) are the 2 nodes (node numbers) defining this element

• cross_section is the cross section area, \([L^2]\)

• elastic_modulus elastic modulus of the material which makes up the beam, \([F/L^2]\)

• shear_modulus shear modulus of the material which makes up the beam, \([F/L^2]\)

• torsion_Jx cross section polar (torsional) moment of inertia, \([L^4]\)

• bending_Iy cross section moment of inertia about local \(y\) axis, \([L^4]\)

• bending_Iz cross section moment of inertia about local \(z\) axis, \([L^4]\)

• mass_density mass per unit volume of the material, \([M/L^3]\)

• xz_plane_vector a vector which defines the orientation of the local (beam coordinate system) \(xz\) plane in global coordinates. **NOTE:** Please make sure that your \(xz\_plane\_vector\) is a bit away from the actual local \(x\) axes, the axes that runs along the beam element, in order to prevent numerical problems that might appear when vector cross products are performed inside the program... It is suggested that your \(xz\_plane\_vector\) be closer to local \(z\) axes... See Figure 205.4 on Page 864 for more in depth explanation of \(xz\_plane\_vector\).

• joint_1_offset vector defining the rigid offset between end of beam and connection node 1, \([L]\)

• joint_2_offset vector defining the rigid offset between end of beam and connection node 2, \([L]\)

Description of output by this element can be found in Section 206.8.4

more on this finite element can be found in Section 102.7 on Page 120 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
Any vector that is in the local xz plane can be used except a vector that is in the direction of the local x axis. Vector that can be used \( \vec{v}_x = (0, 1, 1) \).

\( \vec{v}_x = (0, 1, 0) \) Do not use
\( \vec{v}_x = (0, -1, 0) \) Do not use
205.3.4.53  Modeling, Finite Element: Large Displacement Elastic Beam–Column Element, with Corotational Transformation

The command is:

```plaintext
add element # <.> type beam_elastic_corotational with nodes (<.>, <.>)
cross_section = <L^2>
elastic_modulus = <F/L^2>
shear_modulus = <F/L^2>
torsion_Jx = <length^4>
bending_Iy = <length^4>
bending_Iz = <length^4>
mass_density = <M/L^3>
xz_plane_vector = (<.>, <.>, <.> )
joint_1_offset = (<L>, <L>, <L> )
joint_2_offset = (<L>, <L>, <L> );
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- type beam_elastic is the element type
- with nodes (n1, n2) are the 2 nodes (node numbers) defining this element
- cross_section is the cross section area, \([L^2]\)
- elastic_modulus elastic modulus of the material which makes up the beam, \([F/L^2]\)
- shear_modulus shear modulus of the material which makes up the beam, \([F/L^2]\)
- torsion_Jx cross section polar (torsional) moment of inertia, \([L^4]\)
- bending_Iy cross section moment of inertia about local \(y\) axis, \([L^4]\)
- bending_Iz cross section moment of inertia about local \(z\) axis, \([L^4]\)
- mass_density mass per unit volume of the material, \([M/L^3]\)
- xz_plane_vector a vector which defines the orientation of the local (beam coordinate system) \(xz\) plane in global coordinates.
- joint_1_offset vector defining the rigid offset between end of beam and connection node 1, \([L]\)
- joint_2_offset vector defining the rigid offset between end of beam and connection node 2, \([L]\)
205.3.4.54  Modeling, Finite Element: Timoshenko Elastic Beam–Column Element

The command is:

```plaintext
1 add element # <.> type beam_elastic_Timoshenko with nodes (<.>, <.>)
2    cross_section = <L^2>
3    elastic_modulus = <F/L^2>
4    shear_modulus = <F/L^2>
5    torsion_Jx = <length^4>
6    bending_Iy = <length^4>
7    bending_Iz = <length^4>
8    mass_density = <M/L^3>
9    shear_correction_coefficient = <.>
10   xz_plane_vector = (<.>, <.>, <.> )
11   joint_1_offset = (<L>, <L>, <L> )
12   joint_2_offset = (<L>, <L>, <L> );
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used)

- type beam_elastic is the element type

- with nodes (n1, n2) are the 2 nodes (node numbers) defining this element

- cross_section is the cross section area, [L^2]

- elastic_modulus elastic modulus of the material which makes up the beam, [F/L^2]

- shear_modulus shear modulus of the material which makes up the beam, [F/L^2]

- torsion_Jx cross section polar (torsional) moment of inertia, [L^4]

- bending_Iy cross section moment of inertia about local y axis, [L^4]

- bending_Iz cross section moment of inertia about local z axis, [L^4]

- mass_density mass per unit volume of the material, [M/L^3]

- shear_correction_coefficient a parameter for shear correction. When this parameter becomes very large, the Timoshenko beam element becomes Euler-Bernoulli beam. If not specifically calibrated, can use 1.0 for this parameter.

- xz_plane_vector a vector which defines the orientation of the local (beam coordinate system) xz plane in global coordinates.
• \texttt{joint\_1\_offset} vector defining the rigid offset between end of beam and connection node 1, \([L]\)

• \texttt{joint\_2\_offset} vector defining the rigid offset between end of beam and connection node 2, \([L]\)

Description of output by this element can be found in Section 206.8.4. more on this finite element can be found in Section 102.7 on Page 120 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.55 Modeling, Finite Element: Timoshenko Elastic Beam–Column Element with Directional Shear Correction Coefficients

The command is:

```plaintext
add element # <.> type beam_elastic_Timoshenko_directional with nodes (<.>, <.>)
cross_section = <L^2>
elastic_modulus = <F/L^2>
shear_modulus = <F/L^2>
torsion_Jx = <length^4>
bending_Iy = <length^4>
bending_Iz = <length^4>
mass_density = <M/L^3>
shear_correction_coefficient_y = <.>
shear_correction_coefficient_z = <.>
xz_plane_vector = (<.>, <.>, <.>)
joint_1_offset = (<L>, <L>, <L>)
joint_2_offset = (<L>, <L>, <L>);
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- type beam_elastic is the element type
- with nodes (n1, n2) are the 2 nodes (node numbers) defining this element
- cross_section is the cross section area, \([L^2]\)
- elastic_modulus elastic modulus of the material which makes up the beam, \([F/L^2]\)
- shear_modulus shear modulus of the material which makes up the beam, \([F/L^2]\)
- torsion_Jx cross section polar (torsional) moment of inertia, \([L^4]\)
- bending_Iy cross section moment of inertia about local \(y\) axis, \([L^4]\)
- bending_Iz cross section moment of inertia about local \(z\) axis, \([L^4]\)
- mass_density mass per unit volume of the material, \([M/L^3]\)
- shear_correction_coefficient_y parameter for shear correction about local \(y\) axis. When this parameter becomes very large, the Timoshenko beam element becomes Euler-Bernoulli beam. If not specifically calibrated, can use 1.0 for this parameter.
• **shear_correction_coefficient_z** parameter for shear correction about local \( z \) axis. When this parameter becomes very large, the Timoshenko beam element becomes Euler-Bernoulli beam. If not specifically calibrated, can use 1.0 for this parameter.

• **xz_plane_vector** a vector which defines the orientation of the local (beam coordinate system) \( xz \) plane in global coordinates.

• **joint_1_offset** vector defining the rigid offset between end of beam and connection node 1, \([L]\)

• **joint_2_offset** vector defining the rigid offset between end of beam and connection node 2, \([L]\)

Description of output by this element can be found in Section 206.8.4. More on this finite element can be found in Section 102.7 on Page 120 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.56 Modeling, Finite Element: Adding 1D Fiber to a Beam Cross Section

Fibers can be added to the fiber beam cross section.

The command is:

```
add fiber # <.> using material # <.> to section # <.> fiber_cross_section = <area> fiber_location = (<L>,<L>);
```

For example:

```
add fiber # 1 using material # 1 to section # 1 fiber_cross_section = 5*cm^2 fiber_location = (10*cm,10*cm);
```

adds a fiber number 1 to section number 1 at coordinates $y = 10$ cm, $z = 10$ cm with cross section area of $5 \text{cm}^2$ using material number 1.

The material for fiber must be a uniaxial material, for example uniaxial_concrete02, uniaxial_elastic, uniaxial_steel01, and uniaxial_steel02.
205.3.4.57 Modeling, Finite Element: Adding Fiber Section to the Finite Element Model

Fiber section can be added to the finite element model.

The command is:

```
1 add section # <.> type FiberSection
2 TorsionConstant_GJ = <F*L^2>;
```

where

- $TorsionConstant_GJ$ provides a linear torsional stiffness to the element.

Fibers can be added to the section as described in section 205.3.4.56 on page 870.

The command is:

```
1 add fiber # <.> using material # <.> to section # <.> fiber_cross_section = <area>
2 fiber_location = (<L>,<L>);
```

where

- $fiber_cross_section$ is the area of the fiber element. (Total cross section are is the sum of all fiber areas) $[L^2]$
- $fiber_location$ location of the fiber in the beam local Y-Z plane.
205.3.4.58 Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element

```plaintext
add element # <.> type BeamColumnDispFiber3d with nodes (<.>, <.>)
  number_of_integration_points = <.>
  section_number = <.>
  mass_density = <M/L^3>
  xz_plane_vector = (<.>, <.>, <.> )
  joint_1_offset = (<L>, <L>, <L> )
  joint_2_offset = (<L>, <L>, <L> );
```

where

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- type BeamColumnDispFiber3d is the element type
- with nodes (n1, n2) are the 2 nodes defining this element
- number_of_integration_points is number of integration points to be used along the beam element
- section_number is the number of predefined section
- mass_density mass per unit volume of the material, \([M/L^3]\)
- xz_plane_vector unit vector which defines the orientation of the web of the beam in global coordinates.
- joint_1_offset vector defining the rigid offset between end of beam and connection node 1, \([L]\)
- joint_2_offset vector defining the rigid offset between end of beam and connection node 2, \([L]\)

Description of output by this element can be found in Section 206.8.6,
205.3.4.59 Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element with Co-Rotational Coordinate Transformation

```plaintext
add element # <.> type BeamColumnDispFiber3d_Corotational with nodes (<.>, <.>)
number_of_integration_points = <.>
section_number = <.>
mass_density = <M/L^3>
xz_plane_vector = (<.>, <.>, <.>)
joint_1_offset = (<L>, <L>, <L> )
joint_2_offset = (<L>, <L>, <L> )
```

where

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- type BeamColumnDispFiber3d_Corotational is the element type
- with nodes (n1, n2) are the 2 nodes defining this element
- number_of_integration_points is number of integration points to be used along the beam element
- section_number is the number of predefined section
- mass_density mass per unit volume of the material, [M/L^3]
- xz_plane_vector unit vector which defines the orientation of the web of the beam in global coordinates.
- joint_1_offset vector defining the rigid offset between end of beam and connection node 1, [L]
- joint_2_offset vector defining the rigid offset between end of beam and connection node 2, [L]

Description of output by this element can be found in section 206.8.6.
The co-rotational formulation used in this element is based on Crisfield (1990).
205.3.4.60 Modeling, Finite Element: 3DOF+6DOF=9DOF Beam-Column Element

```plaintext
add element # <.> type beam_9dof_elastic
   with nodes (<.>, <.>)
cross_section = <L^2>
elastic_modulus = <F/L^2>
shear_modulus = <F/L^2>
torsion_Jx = <length^4>
bending_Iy = <length^4>
bending_Iz = <length^4>
mass_density = <M/L^3>
   xz_plane_vector = (<.>, <.>, <.>)
   joint_1_offset = (<L>, <L>, <L>)
   joint_2_offset = (<L>, <L>, <L>);
```

where

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- type beam_9dof_elastic is the element type
- with nodes (n1, n2) are the 2 nodes defining this element, where the first node (n1) is the one with 3 DOFs and the second (n2) is the one with 6 DOFs
- cross_section is the cross section area, \([L^2]\)
- elastic_modulus elastic modulus of the material which makes up the beam, \([F/L^2]\)
- shear_modulus shear modulus of the material which makes up the beam, \([F/L^2]\)
- torsion_Jx cross section polar (torsional) moment of inertia, \([L^4]\)
- bending_Iy cross section moment of inertia about local y axis, \([L^4]\)
- bending_Iz cross section moment of inertia about local z axis, \([L^4]\)
- mass_density mass per unit volume of the material, \([M/L^3]\)
- xz_plane_vector unit vector which defines the orientation of the web of the beam in global coordinates.
- joint_1_offset vector defining the rigid offset between end of beam and connection node 1, \([L]\)
- joint_2_offset vector defining the rigid offset between end of beam and connection node 2, \([L]\)
This finite element has only 3DOFs (translations) at the first node, and full 6DOFs at the other, second node. Due to missing rotational stiffness on first, 3DOF node, this beam has zero torsional stiffness.

This element is useful for connection of solid (3DOFs per node) and structural (6DOFs per node) elements. If this beam element is used on its own, DOF that corresponds to torsion of the second node (DOF number 7), should be fixed as this beam does not provide that stiffness.

More on this finite element can be found in Section 102.8 on Page 123 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.61 Modeling, Finite Element: 4 Node ANDES Shell with Drilling DOFs

ANDES based 3D shell element including drilling degrees of freedom. Made up by patching together 4 ANDES shell triangle elements (and then averaging two and two squares made up two and two triangles).

The command is:

```plaintext
1 add element # <.> type 4NodeShell_ANDES
2    with nodes (<.>, <.>, <.>)
3    use material # <.>
4    thickness = <L> ;
```

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- material # <.> number of a previously defined material. (see add material ...)
- thickness shell thickness, [L]

Description of output by this element can be found in Section 206.8.5. More on this finite element can be found in Section 102.10 on page 129 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.62 Modeling, Finite Element: 3 Node ANDES Shell with Drilling DOFs

```
1  add element # <.> type 3NodeShell_ANDES
2    with nodes (<.>, <.>, <.>)
3  use material # <.>
4  thickness = <L> ;
```

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
205.3.4.63 Modeling, Finite Element: 4 Node Shell NLDKGQ, or 4 Node Shell Xin-Zheng-Lu

This is a 3D quadrilateral shell element with membrane and drill DOFs based on the theory of generalized conforming element. This element accounts for the geometric nonlinearity of large deformation using a simplified version of updated Lagrangian formulation, where nodal coordinates are updated in each step, however strains and stresses are still calculated with reference to the original, undeformed system. It can be used together with elastic or inelastic sections. This element was originally developed by Professor Xin-Zheng Lu (Tsinghua University) and his students.

The command is:

```
1 add element # <.> type 4NodeShell_NLDKGQ
2   with nodes (<., <.>, <.>, <.>)
3   section_number = <.>;
```

It can also be called using the alternative command:

```
1 add element # <.> type 4NodeShell_XinZhengLu_Tsinghua
2   with nodes (<., <.>, <.>, <.>)
3   section_number = <.>;
```

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- with nodes (n1, n2, n3, n4) are the 4 nodes defining this element
- section_number is the number of predefined shell cross section, described on page 879.
### 205.3.4.64 Modeling, Finite Element: Inelastic Layered Shell Section

This command is used to add a layered shell section. The section is made up of a number of layers with different thicknesses and different material properties (i.e., concrete layers or rebar layers). This type of section is used together with plane stress materials and shell elements.

The command is:

```plaintext
1  add section # <.> type LayeredShellFiber
2   number_of_layers = <.>
3   thickness_array = "<.>,<.>..."
4   with material # "<.>,<.>..."
5   thickness_scale_unit = <L>
6   outofplane_shear_modulus = <F/L^2>;
```

where

- `number_of_layers` is the number of layers that the section has
- `thickness_array` is the relative thickness of each layer
- `with material #` is the material tag of each layer, only plane stress materials can be used here, see pages 842 and 843.
- `thickness_scale_unit` is the total thickness of the section
- `outofplane_shear_modulus` is the out-of-plane shear modulus of the section
205.3.4.65  Modeling, Finite Element: ElasticMembranePlaneStress Element (to be removed!)

NOTE: this element is being removed, and will not be available after Real ESSI version 19.07 (current). This is a 2D finite element, and we only maintain 3D finite elements. This element is replaced by a 3D 27 node elastic and/or elastic-plastic wall/plate/shell brick element.

The command is:

```
1 add element No (or #) <element_number>
2   type ElasticMembranePlaneStress
3   with nodes (n1, n2, n3, n4)
4   use material No (or #) <material_number>
5   thickness = <L> ;
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used).
- type ElasticMembranePlaneStress is the element type.
- with nodes (n1, n2, n3, n4) are the 4 nodes (node numbers) defining this element.
- use material No (or #) is the material number for linear elastic material that makes up the element.
- thickness is the thickness of the membrane.
205.3.4.66 Modeling, Finite Element: InelasticMembranePlaneStress Element (to be removed!)

NOTE: this element is being removed, and will not be available after Real ESSI version 19.07 (current). This is a 2D finite element, and we only maintain 3D finite elements. This element is replaced by a 3D 27 node elastic and/or elastic-plastic wall/plate/shell brick element.

The command is:

```
1 add element No (or #) <element_number>
2   type InelasticMembranePlaneStress
3   with nodes (n1, n2, n3, n4)
4   use material No (or #) <material_number>
5   ;
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used).
- type InelasticMembranePlaneStress is the element type.
- with nodes (n1, n2, n3, n4) are the 4 nodes (node numbers) defining this element.
- use material No (or #) is the material number for inelastic material that makes up the element. Since this is a plane stress element, material needs to have plane stress constitutive integration algorithm available. In addition, this material should specify thickness of the element. Different layers and their thicknesses for different materials (for example concrete and steel) will be defined within material definition. PlaneStressLayeredMaterial is a material of this type.
205.3.4.67 Modeling, Finite Element: SuperElementLinearElasticImport

The command is:

```
1 add element No (or #) <element_number> 
  type SuperElementLinearElasticImport 
  with hdf5_file = <string> 
2 ;
```

where

- No (or #)<element_number> is a unique element integer number (does not have to be sequential, any unique positive integer number can be used).
- type SuperElementLinearElasticImport is the element type.
- hdf5_file specifies the HDF5 filename of the SuperElement with SuperElement data. The HDF5 file should contain the following datasets:
  - Node dataset within HDF5 file is organized in a column (a 1D dataset), and it specifies the node tags/numbers of nodes that make up the SuperElement.
  - DofList dataset within HDF5 file is a organized in a column, and it specifies the number of DOFs per each Node. For example if nodes are representing structural elements, they usually have 6 DOFs per node, while solids will have 3 DOFs per node. DofList dataset has to have the same number of entries as Node dataset, as each entry in DofList corresponds to one node from Node dataset.
  - MassMatrix is a matrix, that sets masses/numbers for a mass matrix of the SuperElement.
  - StiffnessMatrix is a matrix, that sets stiffness/numbers for a stiffness matrix of the SuperElement.
  - ConnectNode dataset within HDF5 file is organized in a column (a 1D dataset), and it specifies the node tags/numbers of nodes that are going to be connected to Real-ESSI mesh.
  - ConnectNodeCoordinate dataset within HDF5 file is organized in a matrix (a 2D dataset), and it specifies the nodal coordinates for nodes that are going to be connected to Real-ESSI mesh. Since each node has 3 coordinates, the length of ConnectNodeCoordinate is the same as the length of ConnectNode and each line has three entries, for X, Y and Z coordinates of given node.

In addition to the minimum dataset requirements above, users can get more output from Real-ESSI:
- Results for individual finite elements (internal forces, etc.), can be obtained if node, DofList, mass matrix and stiffness matrix for each finite element within the super element are provided.
- Graphical post-processing can be obtained if coordinates for all nodes and their connectivity into finite elements are provided (a mesh data).
205.3.4.68 Modeling, Finite Element: 8 Node Brick Element

The command is:

```plaintext
1 add element # <element_number> type 8NodeBrick
2 using <.> Gauss points each direction
3 with nodes (n1, n2, n3, n4, n5, n6, n7, n8)
4 use material No (or #) <material_number>;
```

and/or:

```plaintext
1 add element # <element_number> type 8NodeBrick
2 with nodes (n1, n2, n3, n4, n5, n6, n7, n8)
3 use material No (or #) <material_number>;
```

where:

- `No (or #)<element_number>` is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- `type 8NodeBrick` is the element type.
- `with nodes (n1, n2, n3, n4, n5, n6, n7, n8)` are the 8 nodes for this element, in the order as per figure below

![Diagram of 8 Node Brick Element](image)

- `using <.> Gauss points each direction` is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility.

---

For 8 node bricks 2 Gauss points are used in each direction \((2 \times 3 \times 3)\), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction \((3 \times 3 \times 3)\).

- **use material No.** (or #) is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms). Use LT version with LT materials.

Description of output by this element can be found in section 206.8.2.

More on this finite element can be found in Section 102.4.1 on page 108 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.69 Modeling, Finite Element: 20 Node Brick Element

The command is:

```
1 add element No (or #) <element_number> type 20NodeBrick
2   using <.> Gauss points each direction
3   with nodes (n1, n2, n3, n4, n5, n6, n7, n8, 
4       n9, n10, n11, n12, n13, n14, n15, n16,  
5       n17, n18, n19, n20 )
6   use material No (or #) <material_number>;
```

and/or

```
1 add element No (or #) <element_number> type 20NodeBrick
2   with nodes (n1, n2, n3, n4, n5, n6, n7, n8,  
3       n9, n10, n11, n12, n13, n14, n15, n16,  
4       n17, n18, n19, n20 )
5   use material No (or #) <material_number>;
```

where:

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)

- type 20NodeBrick is the element type. 20NodeBrick_elastic can be used if elastic material is used. In this case, the stiffness and mass matrices will not be updated at each step.

- with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20) are the 20 nodes for this element, written in the order defined as per figure below

- using <.> Gauss points each direction is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility. For 8 node bricks 2 Gauss points are used in each direction (2 × 3 × 3), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction (3 × 3 × 3),

- use material No (or #) is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms)

Description of output by this element can be found in Section 206.8.2.
More on this finite element can be found in Section 102.4.3 on page 110 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.70 Modeling, Finite Element: 27 Node Brick Element

The command is:

```plaintext
add element # <element_number>
  type 27NodeBrick
  using <.> Gauss points each direction
  with nodes (n1, n2, n3, n4, n5, n6, n7, n8,
  n9, n10, n11, n12, n13, n14, n15, n16,
  n17, n18, n19, n20, n21, n22, n23,
  n124, n25, n26, n27)
  use material # <material_number>;
```

and/or

```plaintext
add element # <element_number>
  type 27NodeBrick
  with nodes (n1, n2, n3, n4, n5, n6, n7, n8,
  n9, n10, n11, n12, n13, n14, n15, n16,
  n17, n18, n19, n20, n21, n22, n23,
  n124, n25, n26, n27)
  use material # <material_number>;
```

where:

- No (or #)<element_number> is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)

- type 27NodeBrick is the element type.

- with nodes (n1, n2, n3, n4, n5, n6, n7, n8,
  n9, n10, n11, n12, n13, n14, n15, n16,
  n17, n18, n19, n20, n21, n22, n23,
  n124, n25, n26, n27) are the 27 nodes for this element, written in the order defined as per this figure

- using <.> Gauss points each direction is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility. For 8 node bricks 2 Gauss points are used in each direction ($2 \times 3 \times 3$), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction ($3 \times 3 \times 3$).

- use material No (or #) is the material number which makes up the element (nonlinear elastic and/or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms).
Description of output by this element can be found in Section 206.8.2.

More on this finite element can be found in Section 102.4.4 on page 112 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.71 Modeling, Finite Element: Variable 8-27 Node Brick Element

The command is:

```
1 add element No (or #) <element_number> type variable_node_brick_8_to_27
2    using <.> Gauss points each direction
3    with nodes (n1, n2, n3, n4, n5, n6, n7, n8,
4        n9, n10, n11, n12, n13, n14, n15, n16,
5        n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27)
6    use material No (or #) <material_number>;
```

and/or

```
1 add element No (or #) <element_number> type variable_node_brick_8_to_27
2    using <.> Gauss points each direction
3    with nodes (n1, n2, n3, n4, n5, n6, n7, n8,
4        n9, n10, n11, n12, n13, n14, n15, n16,
5        n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27)
6    use material No (or #) <material_number>;
```

where:

- `No (or #) <element_number>` is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)

- `type variable_node_brick_8_to_27` is the element type

- `with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27)` are the 8 to 27 nodes for this element, written in the order defined as per this figure. Nodes 1-8 are obligatory, while any other nodes can be used but do not have to, the element will automatically pick proper shape functions. This element is good for transitions in meshing.

- `using <.> Gauss points each direction` is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility. For 8 node bricks 2 Gauss points are used in each direction (2 × 3 × 3), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction (3 × 3 × 3).

- `material No (or #)` is the material number which makes up the element (nonlinear elastic and/or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms)
Description of output by this element can be found in Section 206.8.2.
205.3.4.72 Modeling, Finite Element: 8 Node Brick u-p Element

The command is:

```plaintext
1 add element # <.> type 8NodeBrick_up
2     using <.> Gauss points each direction
3     with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
4     use material # <.>
5     porosity = <.>
6     alpha = <.>
7     rho_s = <M/L^3>
8     rho_f = <M/L^3>
9     k_x = <L^3*T/M>
10    k_y = <L^3*T/M>
11    k_z = <L^3*T/M>
12    K_s = <F/L^2>
13    K_f = <F/L^2>;
```

and/or

```plaintext
1 add element # <.> type 8NodeBrick_up
2     with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
3     use material # <.>
4     porosity = <.>
5     alpha = <.>
6     rho_s = <M/L^3>
7     rho_f = <M/L^3>
8     k_x = <L^3*T/M>
9     k_y = <L^3*T/M>
10    k_z = <L^3*T/M>
11    K_s = <F/L^2>
12    K_f = <F/L^2>;
```

where:

- No (or #)<element_number> is the unique element integer number that does not have to be sequential, any unique positive integer number can be used.
- type 8NodeBrick_up is the element type/name.
- with nodes (n1, n2, n3, n4, n5, n6, n7, n8) are the 8 nodes for this element, is specified order.
- using <.> Gauss points each direction is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility.
For 8 node bricks 2 Gauss points are used in each direction \((2 \times 3 \times 3)\), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction \((3 \times 3 \times 3)\).

- **use material No.** (or #) is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms). Use LT version with LT materials.

- **porosity** is the porosity \((n = V_{\text{voids}}/V_{\text{total}})\) of material in this element.

- **alpha** is the parameter controlling level of effective stress analysis. For soils, usually \(\alpha = 1\) is used, while for other materials (saturated concrete, bone material, etc.) lower values are used \((0 \leq \alpha \leq 1)\).

- **rho_s** is the density of particles of the solid phase. It is important to note that this is a density of the actual mineral that makes up solid particles!

- **rho_f** is the density of pore fluid. It is usually density of water, however, for unsaturated and partially saturated materials, this density will be different, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

- **k_x** is the permeability in the \(x\) direction (global \(x\)) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **k_y** is the permeability in the \(y\) direction (global \(y\)) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **k_z** is the permeability in the \(z\) direction (global \(z\)) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **K_s** is the bulk modulus of the soil phase particles. It is important to note that this is a bulk modulus of the actual mineral that makes up solid particles!

- **K_f** is the bulk modulus of the fluid phase that is found in porous material pores. It is usually bulk modulus of the fluid (physical value of the bulk modulus of fluid, for example water), however, for unsaturated and partially saturated materials, this density is a density of a mixture, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
Note: the permeability \( k_x, k_y, k_z \) is used with dimensions of \( [\text{length}]^3[\text{time}]/[\text{mass}] \), which is different from the usual soil mechanics convention, where the permeability has the dimension of velocity, i.e. \( [\text{length}]/[\text{time}] \). Their values are related by \( k = K/\rho fg \), where \( g \) is the gravitational acceleration at which the permeability is measured.

More on theory for this finite element can be found in Section 102.12.3.3 on page 150 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Description of output by this element can be found in Section 206.8.2 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.73 Modeling, Finite Element: 20 Node Brick u-p Element

The command is:

```plaintext
add element # <.> type 20NodeBrick_up
    using <.> Gauss points each direction
    with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ← <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
    use material # <.>
    porosity = <.>
    alpha = <.>
    rho_s = <M/L^3>
    rho_f = <M/L^3>
    k_x = <L^3*T/M>
    k_y = <L^3*T/M>
    k_z = <L^3*T/M>
    K_s = <F/L^2>
    K_f = <F/L^2>
```

and/or

```plaintext
add element # <.> type 20NodeBrick_up
    with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ← <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
    use material # <.>
    porosity = <.>
    alpha = <.>
    rho_s = <M/L^3>
    rho_f = <M/L^3>
    k_x = <L^3*T/M>
    k_y = <L^3*T/M>
    k_z = <L^3*T/M>
    K_s = <F/L^2>
    K_f = <F/L^2>
```

where:

- No (or #)<element_number> is the unique element integer number that does not have to be sequential, any unique positive integer number can be used.
- type 8NodeBrick_up is the element type/name.
- with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, ← n16, n17, n18, n19, n20) are the 20 nodes for this element, is specified order.
- using <.> Gauss points each direction is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the
brick finite element (above) without number of Gauss points control is kept for back compatibility. For 8 node bricks 2 Gauss points are used in each direction \((2 \times 3 \times 3)\), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction \((3 \times 3 \times 3)\).

- **use material No** (or #) is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms). Use LT version with LT materials.

- **porosity** is the porosity \((n = V_{voids}/V_{total})\) of material in this element.

- **alpha** is the parameter controlling level of effective stress analysis. For soils, usually \(\alpha = 1\) is used, while for other materials (saturated concrete, bone material, etc.) lower values are used \((0 \leq \alpha \leq 1)\).

- **rho_s** is the density of particles of the solid phase. It is important to note that this is a density of the actual mineral that makes up solid particles!

- **rho_f** is the density of pore fluid. It is usually density of water, however, for unsaturated and partially saturated materials, this density will be different, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

- \(K_x\) is the permeability in the x direction (global x) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- \(k_y\) is the permeability in the y direction (global y) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- \(k_z\) is the permeability in the z direction (global z) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- \(K_s\) is the bulk modulus of the soil phase particles. It is important to note that this is a bulk modulus of the actual mineral that makes up solid particles!

- \(K_f\) is the bulk modulus of the fluid phase that is found in porous material pores. It is usually bulk modulus of the fluid (physical value of the bulk modulus of fluid, for example water), however, for unsaturated and partially saturated materials, this density is a density of a mixture, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
**Note:** the permeability $k_x, k_y, k_z$ is used with dimensions of $[\text{length}]^3[\text{time}]/[\text{mass}]$, which is different from the usual soil mechanics convention, where the permeability has the dimension of velocity, i.e. $[\text{length}]/[\text{time}]$. Their values are related by $k = K/\rho fg$, where $g$ is the gravitational acceleration at which the permeability is measured.

More on theory for this finite element can be found in section 102.12.3.3 on page 150 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.74 Modeling, Finite Element: 27 Node Brick \( u-p \) Element

The command is:

```plaintext
add element # <.> type 27NodeBrick_up
  using <.> Gauss points each direction
  with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, ←
  n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27) use material # <.>
  porosity = <.>
  alpha = <.>
  rho_s = <M/L^3>
  rho_f = <M/L^3>
  k_x = <L^3*T/M>
  k_y = <L^3*T/M>
  k_z = <L^3*T/M>
  K_s = <F/L^2>
  K_f = <F/L^2>
```

and/or

```plaintext
add element # <.> type 27NodeBrick_up
  with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, ←
  n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27) use material # <.>
  porosity = <.>
  alpha = <.>
  rho_s = <M/L^3>
  rho_f = <M/L^3>
  k_x = <L^3*T/M>
  k_y = <L^3*T/M>
  k_z = <L^3*T/M>
  K_s = <F/L^2>
  K_f = <F/L^2>
```

where:

- No (or #)<element_number> is the unique element integer number that does not have to be sequential, any unique positive integer number can be used.

- type 8NodeBrick_up is the element type/name.

- with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, ←
  n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27) are the 27 nodes for this element, is specified order.

- using <.> Gauss points each direction is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can
be from 1 to 6 Gauss points used (uniformly) in each direction \((r_1, r_2, \text{ and } r_3)\). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility. For 8 node bricks 2 Gauss points are used in each direction \((2 \times 3 \times 3)\), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction \((3 \times 3 \times 3)\).

- **use material No (or #)** is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms). Use LT version with LT materials.

- **porosity** is the porosity \((n = V_{\text{voids}}/V_{\text{total}})\) of material in this element.

- **alpha** is the parameter controlling level of effective stress analysis. For soils, usually \(\alpha = 1\) is used, while for other materials (saturated concrete, bone material, etc.) lower values are used \((0 \leq \alpha \leq 1)\).

- **rho_s** is the density of particles of the solid phase. It is important to note that this is a density of the actual mineral that makes up solid particles!

- **rho_f** is the density of pore fluid. It is usually density of water, however, for unsaturated and partially saturated materials, this density will be different, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

- **k_x** is the permeability in the x direction (global x) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **k_y** is the permeability in the y direction (global y) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **k_z** is the permeability in the z direction (global z) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **K_s** is the bulk modulus of the soil phase particles. It is important to note that this is a bulk modulus of the actual mineral that makes up solid particles!

- **K_f** is the bulk modulus of the fluid phase that is found in porous material pores. It is usually bulk modulus of the fluid (physical value of the bulk modulus of fluid, for example water), however, for
unsaturated and partially saturated materials, this density is a density of a mixture, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Note that, the permeability $k$ is used with dimensions of $[\text{length}]^3[\text{time}]/[\text{mass}]$, which is different from the usual soil mechanics convention, where the permeability has the dimension of velocity, i.e. $[\text{length}]/[\text{time}]$. Their values are related by $k = K/\rho_f g$, where $g$ is the gravitational acceleration at which the permeability is measured.

More on theory for this finite element can be found in Section 102.12.3.3 on page 150 of the main document. Description of output by this element can be found in Section 206.8.2.
205.3.4.75 **Modeling, Finite Element: 8 Node Brick u-p-U Element**

The command is:

```plaintext
add element # <.> type 8NodeBrick_upU
   using <.> Gauss points each direction
   with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
   use material # <.>
   porosity = <.>
   alpha = <.>
   rho_s = <M/L^2>
   rho_f = <M/L^2>
   k_x = <L^-3*T/M>
   k_y = <L^-3*T/M>
   k_z = <L^-3*T/M>
   K_s = <F/L^2>
   K_f = <F/L^2>;
```

and/or

```plaintext
add element # <.> type 8NodeBrick_upU
   with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
   use material # <.>
   porosity = <.>
   alpha = <.>
   rho_s = <M/L^2>
   rho_f = <M/L^2>
   k_x = <L^-3*T/M>
   k_y = <L^-3*T/M>
   k_z = <L^-3*T/M>
   K_s = <F/L^2>
   K_f = <F/L^2>;
```

where:

- No (or #)<element_number> is the unique element integer number that does not have to be sequential, any unique positive integer number can be used.

- **type 8NodeBrick_up** is the element type/name.

- **with nodes (n1, n2, n3, n4, n5, n6, n7, n8)** are the 8 nodes for this element, is specified order.

- **using <.> Gauss points each direction** is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility.
For 8 node bricks 2 Gauss points are used in each direction ($2 \times 3 \times 3$), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction ($3 \times 3 \times 3$).

- **use material No (or #)** is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms). Use LT version with LT materials.

- **porosity** is the porosity ($n = \frac{V_{voids}}{V_{total}}$) of material in this element.

- **$\alpha$** is the parameter controlling level of effective stress analysis. For soils, usually $\alpha = 1$ is used, while for other materials (saturated concrete, bone material, etc.) lower values are used ($0 \leq \alpha \leq 1$).

- **$\rho_s$** is the density of particles of the solid phase. It is important to note that this is a density of the actual mineral that makes up solid particles!

- **$\rho_f$** is the density of pore fluid. It is usually density of water, however, for unsaturated and partially saturated materials, this density will be different, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

- **$k_x$** is the permeability in the x direction (global x) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, $k_x = k_y = k_z$.

- **$k_y$** is the permeability in the y direction (global y) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, $k_x = k_y = k_z$.

- **$k_z$** is the permeability in the z direction (global z) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, $k_x = k_y = k_z$.

- **$K_s$** is the bulk modulus of the soil phase particles. It is important to note that this is a bulk modulus of the actual mineral that makes up solid particles!

- **$K_f$** is the bulk modulus of the fluid phase that is found in porous material pores. It is usually bulk modulus of the fluid (physical value of the bulk modulus of fluid, for example water), however, for unsaturated and partially saturated materials, this density is a density of a mixture, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
Note that, the permeability $k$ is used with dimensions of \([\text{length}^3\text{[time]}]/[\text{mass}]\), which is different from the usual soil mechanics convention, where the permeability has the dimension of velocity, i.e. \([\text{length}]/[\text{time}]\). Their values are related by $k = K/\rho fg$, where $g$ is the gravitational acceleration at which the permeability is measured.

Please note that the $u - p - U$ element, and the $u - p - U$ formulation is a dynamic formulation and is meant to be used with dynamic analysis, and not static analysis, so that all the element matrices, as described in theory section, noted below, are developed and used.

More on theory for this finite element can be found in Section 102.12.1.7 on page 146 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL). Description of output by this element can be found in Section 206.8.2 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.76  Modeling, Finite Element: 20 Node Brick u-p-U Element

The command is:

```plaintext
add element # <.> type 20NodeBrick_upU
  using <.> Gauss points each direction
  with nodes (<.>, <.>, <.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>)
  use material # <.>
  porosity = <.>
  alpha = <.>
  rho_s = <M/L^2>
  rho_f = <M/L^2>
  k_x = <L^3*T/M>
  k_y = <L^3*T/M>
  k_z = <L^3*T/M>
  K_s = <F/L^2>
  K_f = <F/L^2>;
```

and/or

```plaintext
add element # <.> type 20NodeBrick_upU
  with nodes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>)
  use material # <.>
  porosity = <.>
  alpha = <.>
  rho_s = <M/L^2>
  rho_f = <M/L^2>
  k_x = <L^3*T/M>
  k_y = <L^3*T/M>
  k_z = <L^3*T/M>
  K_s = <F/L^2>
  K_f = <F/L^2>;
```

where:

- No (or #) <element_number> is the unique element integer number that does not have to be sequential, any unique positive integer number can be used.
- type 8NodeBrick_up is the element type/name.
- with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20) are the 20 nodes for this element, is specified order.
- using <.> Gauss points each direction is the number of Gauss points to be used in each direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can be from 1 to 6 Gauss points used (uniformly) in each direction (r1, r2, and r3). Command for the
brick finite element (above) without number of Gauss points control is kept for back compatibility. For 8 node bricks 2 Gauss points are used in each direction ($2 \times 3 \times 3$), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction ($3 \times 3 \times 3$).

- **use material No (or #)** is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms). Use LT version with LT materials.

- **porosity** is the porosity ($n = V_{voids}/V_{total}$) of material in this element.

- **alpha** is the parameter controlling level of effective stress analysis. For soils, usually $\alpha = 1$ is used, while for other materials (saturated concrete, bone material, etc.) lower values are used ($0 \leq \alpha \leq 1$).

- **rho_s** is the density of particles of the solid phase. It is important to note that this is a density of the actual mineral that makes up solid particles!

- **rho_f** is the density of pore fluid. It is usually density of water, however, for unsaturated and partially saturated materials, this density will be different, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

- **k_x** is the permeability in the $x$ direction (global $x$) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, $k_x = k_y = k_z$.

- **k_y** is the permeability in the $y$ direction (global $y$) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, $k_x = k_y = k_z$.

- **k_z** is the permeability in the $z$ direction (global $z$) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, $k_x = k_y = k_z$.

- **K_s** is the bulk modulus of the soil phase particles. It is important to note that this is a bulk modulus of the actual mineral that makes up solid particles!

- **K_f** is the bulk modulus of the fluid phase that is found in porous material pores. It is usually bulk modulus of the fluid (physical value of the bulk modulus of fluid, for example water), however, for unsaturated and partially saturated materials, this density is a density of a mixture, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
Note that, the permeability $k$ is used with dimensions of $[length]^3[time]/[mass]$, which is different from the usual soil mechanics convention, where the permeability has the dimension of velocity, i.e. $[length]/[time]$. Their values are related by $k = K/\rho fg$, where $g$ is the gravitational acceleration at which the permeability is measured.

Please note that the $u - p - U$ element, and the $u - p - U$ formulation is a dynamic formulation and is meant to be used with dynamic analysis, and not static analysis, so that all the element matrices, as described in theory section, noted below, are developed and used.

More on theory for this finite element can be found in section 102.12.1.8 on page 146 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL). Description of output by this element can be found in Section 206.8.2.
205.3.4.77 Modeling, Finite Element: 27 Node Brick \(u-p-U\) Element

The command is:

```plaintext
add element # <.> type 27NodeBrick_upU
  using <.> Gauss points each direction
  with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ←
  <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
  use material # <.>
  porosity = <.>
  alpha = <.>
  rho_s = \(<M/L^2>\)
  rho_f = \(<M/L^2>\)
  k_x = \(<L^3*T/M>\)
  k_y = \(<L^3*T/M>\)
  k_z = \(<L^3*T/M>\)
  K_s = \(<F/L^2>\)
  K_f = \(<F/L^2>\);
```

and/or

```plaintext
add element # <.> type 27NodeBrick_upU
  with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ←
  <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
  use material # <.>
  porosity = <.>
  alpha = <.>
  rho_s = \(<M/L^2>\)
  rho_f = \(<M/L^2>\)
  k_x = \(<L^3*T/M>\)
  k_y = \(<L^3*T/M>\)
  k_z = \(<L^3*T/M>\)
  K_s = \(<F/L^2>\)
  K_f = \(<F/L^2>\);
```

where:

- No (or #)<element_number> is the unique element integer number that does not have to be sequential, any unique positive integer number can be used.

- type 8NodeBrick_up is the element type/name.

- with nodes (n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, ←
  n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27) are the 27 nodes for
  this element, is specified order.

- using <.> Gauss points each direction is the number of Gauss points to be used in each
direction (r1, r2, and r3) for integration of finite element matrices (mass and stiffness). There can
be from 1 to 6 Gauss points used (uniformly) in each direction \((r_1, r_2, \text{ and } r_3)\). Command for the brick finite element (above) without number of Gauss points control is kept for back compatibility. For 8 node bricks 2 Gauss points are used in each direction \((2 \times 3 \times 3)\), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction \((3 \times 3 \times 3)\).

- **use material No (or #)** is the material number which makes up the element (nonlinear elastic or elastic-plastic material properties for each integration (Gauss) point will evolve independently as the element deforms). Use LT version with LT materials.

- **porosity** is the porosity \(n = V_{\text{voids}}/V_{\text{total}}\) of material in this element.

- **\(\alpha\)** is the parameter controlling level of effective stress analysis. For soils, usually \(\alpha = 1\) is used, while for other materials (saturated concrete, bone material, etc.) lower values are used \((0 \leq \alpha \leq 1)\).

- **\(\rho_s\)** is the density of particles of the solid phase. It is important to note that this is a density of the actual mineral that makes up solid particles!

- **\(\rho_f\)** is the density of pore fluid. It is usually density of water, however, for unsaturated and partially saturated materials, this density will be different, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

- **\(k_x\)** is the permeability in the \(x\) direction (global \(x\)) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **\(k_y\)** is the permeability in the \(y\) direction (global \(y\)) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **\(k_z\)** is the permeability in the \(z\) direction (global \(z\)) of the element. It is also important to note about the units used for permeability, as noted below. With isotropic permeability, usually the case, \(k_x = k_y = k_z\).

- **\(K_s\)** is the bulk modulus of the soil phase particles. It is important to note that this is a bulk modulus of the actual mineral that makes up solid particles!

- **\(K_f\)** is the bulk modulus of the fluid phase that is found in porous material pores. It is usually bulk modulus of the fluid (physical value of the bulk modulus of fluid, for example water), however, for
unsaturated and partially saturated materials, this density is a density of a mixture, as described in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Note that, the permeability $k$ is used with dimensions of $[\text{length}]^3[\text{time}]/[\text{mass}]$, which is different from the usual soil mechanics convention, where the permeability has the dimension of velocity, i.e. $[\text{length}]/[\text{time}]$. Their values are related by $k = K/\rho fg$, where $g$ is the gravitational acceleration at which the permeability is measured.

Please note that the $u - p - U$ element, and the $u - p - U$ formulation is a dynamic formulation and is meant to be used with dynamic analysis, and not static analysis, so that all the element matrices, as described in theory section, noted below, are developed and used.

More on theory for this finite element can be found in Section 102.12.1.9 on page 146 of the main document. Description of output by this element can be found in Section 206.8.2.
205.3.4.78 Modeling, Finite Element: 8 Node Cosserat Brick Element

The command is:

```
1 add element # <element_number> type Cosserat8NodeBrick
2   with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
3   use material # <.>
```

where:

- `<element_number>` is the unique element integer number (does not have to be sequential, any unique positive integer number can be used)
- `type Cosserat8NodeBrick` is the element type.
- `with nodes (n1, n2, n3, n4, n5, n6, n7, n8)` are the 8 nodes for this element. Each node should have 6 DOFs for this element. The element should be in the order as per figure below

![Diagram of 8 Node Cosserat Brick Element](image)

- `use material No (or #)` is the material number which makes up the element. The element can use materials Cosserat_linear_elastic_isotropic_3d and Cosserat_von_Mises.
205.3.4.79 Modeling, Finite Element: Bonded Contact/Interface/Joint Element

The command is:

```plaintext
1 add element # <.> type BondedContact
2   with nodes (<.>, <.>)
3   penalty_stiffness = <F/L>
```

where

- `penalty_stiffness` represents the penalty stiffness in the three orthogonal $x$, $y$ and $z$ directions, that connects two nodes of this element.
205.3.4.8 Modeling, Finite Element: Force Based Dry Hard Contact/Interface/Joint Element

The command is:

```plaintext
add element # <.> type ForceBasedHardContact
  with nodes (<.>, <.>)
  axial_stiffness = <F/L>
  shear_stiffness = <F/L>
  axial_viscous_damping = <F/L>
  shear_viscous_damping = <F/L>
  friction_ratio = <.>
  contact_plane_vector = (<.>, <.>, <.>)
```

The axial force $F_a$ and axial stiffness $E_a$ in defined as

$$F_a = E_a \times \delta_a$$  \hspace{1cm} (205.2)

where

- $\delta_a$ refers to the axial relative displacement in axial contact/interface/joint direction,
- $E_a$ refers to the axial stiffness in axial contact direction, and

- `axial_stiffness` (b) represents the stiffness in the axial/axial direction (local $x$ axis).
- `shear_stiffness` is the stiffness in the tangential (shear, local $y$ or $z$ axis) directions.
- `axial_viscous_damping` is the viscous damping in axial/axial.
- `shear_viscous_damping` is the viscous damping in shear.
- `friction_ratio` Coulomb friction ratio.
- `contact_plane_vector` Vector defining the normal to the contact/interface/joint plane.

**IMPORTANT NOTE No. 1:** `contact_plane_vector` defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.81 Modeling, Finite Element: Force Based Dry Soft Contact/Interface/Joint Element

The command is:

```
add element # <.> type ForceBasedSoftContact
  with nodes (<.>, <.>)
  initial_axial_stiffness = <F/L>
  stiffening_rate = <1/m>
  max_axial_stiffness = <F/L>
  shear_stiffness = <F/L>
  axial_viscous_damping = <F/L>
  shear_viscous_damping = <F/L>
  friction_ratio = <.>
  contact_plane_vector = (<.>, <.>, <.>);
```

The axial force $F_a$ and axial stiffness $E_a$ are defined as

\[
F_a = b \times exp(a \times \delta_a) \times \delta_a
\]

\[
E_a = \max(b \times exp(a \times \delta_a) \times (1 + a \times \delta_a), E_{max})
\]  \hspace{1cm} (205.3)

where

- $\delta_a$ refers to the axial relative displacement in axial contact/interface/joint direction,
- $b$ refers to the axial stiffness in axial contact/interface/joint direction,
- $a$ refers to the stiffening rate in axial contact/interface/joint direction,
- $E_{max}$ refers to the maximum axial stiffness, and

- initial_axial_stiffness ($b$) represents the stiffness in the axial direction (local $x$ axis).
- stiffening_rate ($a$) represents exponential stiffening rate $exp(a \times \delta_a)$ in axial direction.
- max_axial_stiffness ($E_{max}$) defines the maximum stiffness in the axial direction (local $x$ axis).
- shear_stiffness is the stiffness in the tangential (shear, local $y$ or $z$ axis) directions.
- axial_viscous_damping is the viscous damping in axial.
- shear_viscous_damping is the viscous damping in shear.
- friction_ratio is Coulomb friction ratio.
- contact_plane_vector is the vector defining the normal to the contact/interface/joint plane.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.
IMPORTANT NOTE No. 2: Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.82 Modeling, Finite Element: Force Based Coupled Hard Contact/Interface/Joint Element

The command is:

```
add element # <.> type ForceBasedCoupledHardContact
  with nodes (<.>, <.>)
  axial_stiffness = <F/L>
  axial_penalty_stiffness = <F/L>
  shear_stiffness = <F/L>
  axial_viscous_damping = <F/L>
  shear_viscous_damping = <F/L>
  friction_ratio = <.>
  contact_plane_vector = (<.>, <.>, <.>);
```

The axial force $F_a$ and axial stiffness $E_a$ are defined as

$$F_a = b \ast \delta_a$$

$$E_a = b$$

where

- $\delta_a$ refers to the axial relative displacement in axial contact/interface/joint direction,
- $b$ refers to the axial stiffness in axial contact/interface/joint direction, and

- **axial_stiffness** ($b$) represents the axial stiffness in the axial direction (local $x$ axis).
- **axial_penalty_stiffness** ($E_p$) defines the penalty stiffness between upU_U and u_u dof to enforce undrained condition in contact/interface/joint axial direction.
- **shear_stiffness** is the stiffness in the tangential (shear, local $y$ or $z$ axis) directions.
- **axial_viscous_damping** is the viscous damping in axial.
- **shear_viscous_damping** is the viscous damping in shear.
- **friction_ratio** Coulomb friction ratio.
- **contact_plane_vector** Vector defining the normal to the contact/interface/joint plane.

**IMPORTANT NOTE No. 1:** **contact_plane_vector** defines a direction from Node I to Node J, that is, **from the first to the second node**. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.
Description of output by this element can be found in Section 206.8.7.
205.3.4.83 Modeling, Finite Element: Force Based Coupled Soft Contact/Interface/Joint Element

The command is:

```plaintext
1 add element # <.> type ForceBasedCoupledSoftContact
2     with nodes (<.>, <.>)
3     initial_axial_stiffness = <F/L>
4     stiffening_rate = <1/m>
5     max_axial_stiffness = <F/L>
6     axial_penalty_stiffness = <F/L>
7     shear_stiffness = <F/L>
8     axial_viscous_damping = <F/L>
9     shear_viscous_damping = <F/L>
10    friction_ratio = <.>
11    contact_plane_vector = (<.>, <.>, <.>);
```

The axial force $F_a$ and axial stiffness $E_a$ are defined as

$$
F_a = b \ast \exp(a \ast \delta_a) \ast \delta_a
$$

$$
E_a = \max(b \ast \exp(a \ast \delta_a) \ast (1 + a \ast \delta_a), E_{max})
$$

(205.5)

where

- $\delta_a$ refers to the axial relative displacement in axial contact/interface/joint direction,
- $b$ refers to the axial stiffness in axial contact/interface/joint direction,
- $a$ refers to the stiffening rate in axial contact/interface/joint direction,
- $E_{max}$ refers to the maximum axial stiffness, and

- `initial_axial_stiffness` ($b$) represents the stiffness in the axial direction (local $x$ axis).
- `stiffening_rate` ($a$) represents exponential stiffening rate $\exp(a \ast \delta_a)$ in axial direction.
- `max_axial_stiffness` ($E_{max}$) defines the maximum stiffness in the axial direction (local $x$ axis).
- `axial_penalty_stiffness` ($E_p$) defines the penalty stiffness between up$U\_U$ and u$u$ dof to enforce undrained condition in contact/interface/joint axial direction.
- `shear_stiffness` is the stiffness in the tangential (shear, local $y$ or $z$ axis) directions.
- `axial_viscous_damping` is the viscous damping in axial.
- `shear_viscous_damping` is the viscous damping in shear.
- `friction_ratio` is Coulomb friction ratio.
- `contact_plane_vector` Vector defining the normal to the contact/interface/joint plane.

**IMPORTANT NOTE No. 1:** `contact_plane_vector` defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.84 Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior

The command is:

```plaintext
add element # <.> type StressBasedHardContact_ElPPlShear
  with nodes (<.>, <.>)
  axial_stiffness = <Pa>
  initial_shear_stiffness = <Pa>
  axial_viscous_damping = <Pa*s>
  shear_viscous_damping = <Pa*s>
  residual_friction_coefficient = <.>
  shear_zone_thickness = <m>
  contact_plane_vector = (<.>, <.>, <.>);
```

and/or:

```plaintext
add element # <.> type StressBasedHardContact_ElPPlShear
  with nodes (<.>, <.>)
  axial_stiffness = <Pa>
  initial_shear_stiffness = <Pa>
  axial_viscous_damping = <Pa*s>
  shear_viscous_damping = <Pa*s>
  residual_friction_coefficient = <.>
  shear_zone_thickness = <m>
  surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$
\sigma_a = b \epsilon_a
$$

$$
E_a = b
$$

where

$E_a = b$ refers to the axial stiffness in axial contact/interface/joint direction,

$\epsilon_a$ refers to the axial strain in axial contact/interface/joint direction $\epsilon_a = \delta_a/h$,

$\delta_a$ is the relative axial penetration in contact axial direction,

$h$ is the shear zone thickness, and

- `axial_stiffness` (b) represents the stiffness in the axial direction (local x axis).
- `initial_shear_stiffness` ($E_s$) is the stiffness in the tangential (shear, local y or z axis) directions at 101kPa axial stress described in Section 104.7.3.2
- `axial_viscous_damping` is the viscous damping in axial.
- `shear_viscous_damping` is the viscous damping in shear.

- `residual_friction_coefficient (\(\mu_r\))` is the residual friction coefficient described in Section 104.7.3.2.

- `shear_zone_thickness \(h\)` is the shear zone thickness.

- `contact_plane_vector` Vector defining the normal to the contact/interface/joint plane.

- `surface_vector_relative_tolerance` defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** `contact_plane_vector` defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.85 Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior

The command is:

```plaintext
add element # <.> type StressBasedHardContact_NonLinHardShear
   with nodes (<>, <>)
   axial_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   residual_friction_coefficient = <.>
   shear_zone_thickness = <m>
   contact_plane_vector = (<>, <>, <>);
```

and/or;

```plaintext
add element # <.> type StressBasedHardContact_NonLinHardShear
   with nodes (<>, <>)
   axial_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   residual_friction_coefficient = <.>
   shear_zone_thickness = <m>
   surface_vector_relative_tolerance = <.>
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$\sigma_a = b \ast \epsilon_a$$
$$E_a = b$$

(205.7)

where

$E_a = b$ refers to the axial stiffness in axial contact/interface/joint direction,

$\epsilon_a$ refers to the axial strain in axial contact direction $\epsilon_a = \delta_a / h$,

$\delta_a$ is the relative axial penetration in contact/interface/joint axial direction,

$h$ is the shear zone thickness, and

- `axial_stiffness` (b) represents the stiffness in the axial direction (local x axis).
- `initial_shear_stiffness` ($E_s$) is the stiffness in the tangential (shear, local y or z axis) directions at 101kPa axial stress, described in Section 104.7.3.4
- `axial_viscous_damping` is the viscous damping in axial.
• shear_viscous_damping is the viscous damping in shear.

• residual_friction_coefficient ($\mu_r$) is the residual frictional parameter as described in Section 104.7.3.3

• shear_zone_thickness $h$ is the shear zone thickness.

• contact_plane_vector Vector defining the normal to the contact/interface/joint plane.

• surface_vector_relative_tolerance defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

IMPORTANT NOTE No. 1: contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

IMPORTANT NOTE No. 2: Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.86 Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior

The command is:

```plaintext
add element # <.> type StressBasedHardContact_NonLinHardSoftShear
  with nodes (<.>, <.>)
  axial_stiffness = <Pa>
  initial_shear_stiffness = <Pa>
  rate_of_softening = <>
  size_of_peak_plateau = <>
  axial_viscous_damping = <Pa*s>
  shear_viscous_damping = <Pa*s>
  peak_friction_coefficient_limit = <>
  peak_friction_coefficient_rate_of_decrease = <.>
  residual_friction_coefficient = <.>
  shear_zone_thickness = <m>
  contact_plane_vector = (<.>, <.>, <.>);
```

and/or:

```plaintext
add element # <.> type StressBasedHardContact_NonLinHardSoftShear
  with nodes (<.>, <.>)
  axial_stiffness = <Pa>
  initial_shear_stiffness = <Pa>
  rate_of_softening = <>
  size_of_peak_plateau = <>
  axial_viscous_damping = <Pa*s>
  shear_viscous_damping = <Pa*s>
  peak_friction_coefficient_limit = <>
  peak_friction_coefficient_rate_of_decrease = <.>
  residual_friction_coefficient = <.>
  shear_zone_thickness = <m>
  surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$\sigma_a = b * \epsilon_a$$

$$E_a = b$$

(205.8)

where

$E_a = b$ refers to the axial stiffness in axial contact/interface/joint direction,

$\epsilon_a$ refers to the axial strain in axial contact direction $\epsilon_a = \delta_a / h$,

$\delta_a$ is the relative axial penetration in contact/interface/joint axial direction,

$h$ is the shear zone thickness, and

- `axial_stiffness` (b) represents the stiffness in the axial direction (local $x$ axis).
• **initial_shear_stiffness** ($E_s$) is the stiffness in the tangential (shear, local $y$ or $z$ axis) directions at 101 kPa axial stress, described in Section 104.7.3.4

• **rate_of_softening** ($R_s$) is the parameter to control the rate of frictional softening described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as

$$
\Delta \mu = -\left(\frac{n \ast R_s (\mu_p - \mu_r)}{(\pi/2)^n \theta^{1/n - 1}}\right) \ast \cos^2 \theta \Delta \gamma^p
$$

(205.9)

$$
\theta = \frac{\mu_p - \mu}{\mu_p - \mu_r} (\pi/2)^n
$$

(205.10)

where, $R_s$ is the frictional softening rate parameter, $\Delta \gamma^p$ is the plastic shear strain and $n$ represents the size of the peak plateau.

$$
\Delta \gamma^p = \sqrt{\Delta \epsilon_{ij}^p \Delta \epsilon_{ij}^p}
$$

(205.11)

• **size_of_peak_plateau** ($n$) is the frictional softening parameter to control the size of plateau as described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as shown in Equation 205.17.

• **axial_viscous_damping** is the viscous damping in axial.

• **shear_viscous_damping** is the viscous damping in shear.

• **peak_friction_coefficient_limit** ($\mu_p$) is the limit to the peak frictional hardening parameter $\mu_p$.

• **peak_friction_coefficient_rate_of_decrease** ($k$) is the rate of decrease of peak frictional hardening parameter $\mu_p$ with axial stress, described in Section 104.7.3.4

$$
\mu_p = \max(\mu_p0, \mu_p0 - k \ast \log(\sigma_a/P_0))
$$

(205.12)

where $\mu_p0$ is the peak frictional hardening limit, $k$ is the peak frictional parameter rate of decrease and $P_0$ is the reference stress of $P_0 = 101 kPa$.

• **residual_friction_coefficient** ($\mu_r$) is the residual frictional parameter as described in Section 104.7.3.4

• **shear_zone_thickness** $h$ is the shear zone thickness.

• **contact_plane_vector** Vector defining the normal to the contact/interface/joint plane.
• `surface_vector_relative_tolerance` defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** `contact_plane_vector` defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.87 Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior

The command is:

```plaintext
1. add element # <.> type StressBasedSoftContact_ElPPlShear
2. with nodes (<.>, <.>)
3. initial_axial_stiffness = <Pa>
4. stiffening_rate = <>
5. max_axial_stiffness = <Pa>
6. initial_shear_stiffness = <Pa>
7. axial_viscous_damping = <Pa*s>
8. shear_viscous_damping = <Pa*s>
9. residual_friction_coefficient = <.>
10. shear_zone_thickness = <m>
11. contact_plane_vector = (<.>, <.>, <.>);
```

and/or:

```plaintext
1. add element # <.> type StressBasedSoftContact_ElPPlShear
2. with nodes (<.>, <.>)
3. initial_axial_stiffness = <Pa>
4. stiffening_rate = <>
5. max_axial_stiffness = <Pa>
6. initial_shear_stiffness = <Pa>
7. axial_viscous_damping = <Pa*s>
8. shear_viscous_damping = <Pa*s>
9. residual_friction_coefficient = <.>
10. shear_zone_thickness = <m>
11. surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$\begin{align*}
\sigma_a &= b \times \exp(a \times \epsilon_a) \times \epsilon_a \\
E_a &= \max\left(b \times \exp(a \times \epsilon_a) \times (1 + a \times \epsilon_a), E_{\text{max}}\right)
\end{align*}$$

(205.13)

where

- $b$ refers to the initial axial stiffness in axial contact/interface/joint direction,
- $a$ refers to the stiffening rate in axial contact direction,
- $E_{\text{max}}$ refers to the maximum axial stiffness,
- $E_a$ refers to the axial stiffness,
- $\epsilon_a$ refers to the axial strain in axial contact/interface/joint direction $\epsilon_a = \delta_a/h$,
- $\delta_a$ is the relative axial penetration in contact axial direction,
- $h$ is the shear zone thickness, and
- initial_axial_stiffness \( b \) represents the stiffness in the axial direction (local \( x \) axis).
- stiffening_rate \( a \) Represents exponential stiffening rate \( e^{a \epsilon} \) in axial direction.
- max_axial_stiffness \( E_{max} \) Defines the maximum stiffness in the axial direction (local \( x \) axis) for the contact/interface/joint element.
- initial_shear_stiffness \( E_s \) Is the stiffness in the tangential (shear, local \( y \) or \( z \) axis) directions at 101kPa axial stress described in Section 104.7.3.2
- axial_viscous_damping Is the viscous damping in axial.
- shear_viscous_damping Is the viscous damping in shear.
- residual_friction_coefficient \( \mu_r \) Is the residual friction coefficient described in Section 104.7.3.2
- shear_zone_thickness \( h \) Is the shear zone thickness
- contact_plane_vector Vector defining the normal to the contact/interface/joint plane.
- surface_vector_relative_tolerance defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.88 Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior

The command is:

```plaintext
add element # <.> type StressBasedSoftContact_NonLinHardShear
    with nodes (<., <.>)
    initial_axial_stiffness = <Pa>
    stiffening_rate = <>
    max_axial_stiffness = <Pa>
    initial_shear_stiffness = <Pa>
    axial_viscous_damping = <Pa*s>
    shear_viscous_damping = <Pa*s>
    residual_friction_coefficient = <.>
    shear_zone_thickness = <m>
    contact_plane_vector = (<., <.>, <.>);
```

and/or:

```plaintext
add element # <.> type StressBasedSoftContact_NonLinHardShear
    with nodes (<., <.>)
    initial_axial_stiffness = <Pa>
    stiffening_rate = <>
    max_axial_stiffness = <Pa>
    initial_shear_stiffness = <Pa>
    axial_viscous_damping = <Pa*s>
    shear_viscous_damping = <Pa*s>
    residual_friction_coefficient = <.>
    shear_zone_thickness = <m>
    surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

\[
\sigma_a = b \ast exp(a \ast \epsilon_a) \ast \epsilon_a \\
E_a = \max(b \ast exp(a \ast \epsilon_a) \ast (1 + a \ast \epsilon_a), E_{max})
\]  

(205.14)

where

- $b$ refers to the initial axial stiffness in axial contact/interface/joint direction,
- $a$ refers to the stiffening rate in axial contact direction,
- $E_{max}$ refers to the maximum axial stiffness,
- $E_a$ refers to the axial stiffness,
- $\epsilon_a$ refers to the axial strain in axial contact/interface/joint direction $\epsilon_a = \delta_a / h$,
- $\delta_a$ is the relative axial penetration in contact axial direction,
- $h$ is the shear zone thickness, and
• **initial_axial_stiffness** \((b)\) represents the stiffness in the axial direction (local \(x\) axis) for 1m penetration.

• **stiffening_rate** \((a)\) Represents exponential stiffening rate \(e^{sr \times \epsilon_a}\) in axial direction.

• **max_axial_stiffness** \((E_{\text{max}})\) Defines the maximum stiffness in the axial direction (local \(x\) axis) for the contact/interface/joint element.

• **initial_shear_stiffness** \((E_s)\) Is the stiffness in the tangential (shear, local \(y\) or \(z\) axis) directions at 101kPa axial stress described in Section 104.7.3.3

• **axial_viscous_damping** Is the viscous damping in axial.

• **shear_viscous_damping** Is the viscous damping in shear.

• **residual_friction_coefficient** \((\mu_r)\) Is the residual frictional parameter as described in Section 104.7.3.3

• **shear_zone_thickness** \(h\) Is the shear zone thickness

• **contact_plane_vector** Vector defining the normal to the contact/interface/joint plane.

• **surface_vector_relative_tolerance** defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.89 Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior

The command is:

```
add element # <.> type StressBasedSoftContact_NonLinHardSoftShear
   with nodes (<.>, <.>)
   initial_axial_stiffness = <Pa>
   stiffening_rate = <>
   max_axial_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   rate_of_softening = <>
   size_of_peak_plateau = <>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   peak_friction_coefficient_limit = <>
   peak_friction_coefficient_rate_of_decrease = <.>
   residual_friction_coefficient = <.>
   shear_zone_thickness = <m>
   contact_zone_thickness = (<.>, <.>, <.>);
```

and/or;

```
add element # <.> type StressBasedSoftContact_NonLinHardSoftShear
   with nodes (<.>, <.>)
   initial_axial_stiffness = <Pa>
   stiffening_rate = <>
   max_axial_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   rate_of_softening = <>
   size_of_peak_plateau = <>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   peak_friction_coefficient_limit = <>
   peak_friction_coefficient_rate_of_decrease = <.>
   residual_friction_coefficient = <.>
   shear_zone_thickness = <m>
   surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ are defined as

$$\sigma_a = b \cdot \exp(a \cdot \epsilon_a) \cdot \epsilon_a$$
$$E_a = \max(b \cdot \exp(a \cdot \epsilon_a) \cdot (1 + a \cdot \epsilon_a), E_{\max})$$

(205.15)

where

$b$ refers to the initial axial stiffness in axial contact/interface/joint direction,

$a$ refers to the stiffening rate in axial contact direction,

$E_{\max}$ refers to the maximum axial stiffness,
$E_a$ refers to the axial stiffness,

$\epsilon_a$ refers to the axial strain in axial contact/interface/joint direction $\epsilon_a = \delta_a / h$,

$\delta_a$ is the relative axial penetration in contact axial direction,

$h$ is the shear zone thickness, and

- **initial_axial_stiffness** \((b)\) represents the stiffness in the axial direction (local \(x\) axis) for 1m penetration.

- **stiffening_rate** \((a)\) Represents exponential stiffening rate $\exp(sr * \epsilon_n)$ in axial direction.

- **max_axial_stiffness** \((E_{max})\) Defines the maximum stiffness in the axial direction (local \(x\) axis) for the contact/interface/joint element.

- **initial_shear_stiffness** \((E_s)\) Is the stiffness in the tangential (shear, local \(y\) or \(z\) axis) directions at 101kPa axial stress, described in Section 104.7.3.4

- **rate_of_softening** \((R_s)\) Is the parameter to control the rate of frictional softening described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power \(n\) with incremental form as

$$
\Delta\mu = -\frac{n * R_s(\mu_p - \mu_r)}{(\pi/2)^n \theta^{1/n-1}} * \cos^2\theta \Delta\gamma^p
$$

$$
\theta = \frac{\mu_p - \mu}{\mu_p - \mu_r}(\pi/2)^n
$$

(205.16)

(205.17)

where, \(R_s\) is the frictional softening rate parameter, \(\Delta\gamma^p\) is the plastic shear strain and \(n\) represents the size of the peak plateau.

$$
\Delta\gamma^p = \sqrt{\Delta\epsilon^p_{ij} \Delta\epsilon^p_{ij}}
$$

(205.18)

- **size_of_peak_plateau** \((n)\) Is the frictional softening parameter to control the size of plateau as described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power \(n\) with incremental form as shown in Equation 205.17.

- **axial_viscous_damping** Is the viscous damping in axial.

- **shear_viscous_damping** Is the viscous damping in shear.

- **peak_friction_coefficient_limit** \((\mu_{p0})\) Is the limit to the peak frictional hardening parameter $\mu_p$. 

• **peak_friction_coefficient_rate_of_decrease** ($k$) is the rate of decrease of peak frictional hardening parameter $\mu_p$ with axial stress, described in Section 104.7.3.4

$$
\mu_p = \max(\mu_{p0}, \mu_{p0} - k \cdot \log(\sigma_a/P_0))
$$

(205.19)

where $\mu_{p0}$ is the peak frictional hardening limit, $k$ is the peak frictional parameter rate of decrease and $P_0$ is the reference stress of $P_0 = 101\, kPa$.

• **residual_friction_coefficient** ($\mu_r$) is the residual frictional parameter as described in Section 104.7.3.4

• **shear_zone_thickness** $h$ is the shear zone thickness

• **contact_plane_vector** Vector defining the normal to the contact/interface/joint plane.

• **surface_vector_relative_tolerance** defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.90 Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior

The command is:

```
add element # <.> type StressBasedCoupledHardContact_ElPPlShear
  with nodes (<.>, <.>)
  axial_stiffness = <Pa>
  axial_penalty_stiffness = <Pa>
  initial_shear_stiffness = <Pa>
  axial_viscous_damping = <Pa*s>
  shear_viscous_damping = <Pa*s>
  residual_friction_coefficient = <.>
  shear_zone_thickness = <m>
  contact_plane_vector = (<.>, <.>, <.>);
```

and/or:

```
add element # <.> type StressBasedCoupledHardContact_ElPPlShear
  with nodes (<.>, <.>)
  axial_stiffness = <Pa>
  axial_penalty_stiffness = <Pa>
  initial_shear_stiffness = <Pa>
  axial_viscous_damping = <Pa*s>
  shear_viscous_damping = <Pa*s>
  residual_friction_coefficient = <.>
  shear_zone_thickness = <m>
  surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

\[
\sigma_a = b \cdot \epsilon_a \\
E_a = b
\]  \hspace{1cm} (205.20)

where

$E_a = b$ refers to the axial stiffness in axial contact/interface/joint direction,

$\epsilon_a$ refers to the axial strain in axial contact direction $\epsilon_a = \delta_a/h$,

$\delta_a$ is the relative axial penetration in contact/interface/joint axial direction,

$h$ is the shear zone thickness, and

- **axial_stiffness** ($b$) represents the stiffness in the axial direction (local $x$ axis).
- **axial_penalty_stiffness** ($E_p$) Defines the axial penalty stiffness between upU_U and u_u dof to enforce undrained condition in contact/interface/joint axial direction.
• initial_shear_stiffness \( (E_s) \) is the stiffness in the tangential (shear, local y or z axis) directions at 101kPa axial stress described in Section 104.7.3.2

• axial_viscous_damping is the viscous damping in axial.

• shear_viscous_damping is the viscous damping in shear.

• residual_friction_coefficient \( (\mu_r) \) is the residual friction coefficient described in Section 104.7.3.2

• shear_zone_thickness \( h \) is the shear zone thickness

• contact_plane_vector Vector defining the normal to the contact/interface/joint plane.

• surface_vector_relative_tolerance defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.91 Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior

The command is:

```plaintext
1  add element # <.> type StressBasedCoupledHardContact_NonLinHardShear
2    with nodes (<.>, <.>)
3    axial_stiffness = <Pa>
4    axial_penalty_stiffness = <Pa>
5    initial_shear_stiffness = <Pa>
6    axial_viscous_damping = <Pa*s>
7    shear_viscous_damping = <Pa*s>
8    residual_friction_coefficient = <.>
9    shear_zone_thickness = <m>
10   contact_plane_vector = (<.>, <.>, <.>);

and/or:

```plaintext
1  add element # <.> type StressBasedCoupledHardContact_NonLinHardShear
2    with nodes (<.>, <.>)
3    axial_stiffness = <Pa>
4    axial_penalty_stiffness = <Pa>
5    initial_shear_stiffness = <Pa>
6    axial_viscous_damping = <Pa*s>
7    shear_viscous_damping = <Pa*s>
8    residual_friction_coefficient = <.>
9    shear_zone_thickness = <m>
10   surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ are defined as

\[
\sigma_a = b \times \epsilon_a = b \times \frac{\delta_a}{h}
\]

\[
E_a = b
\]

where $E_a = b$ refers to the axial stiffness in axial contact/interface/joint direction, $\epsilon_a$ refers to the axial strain in axial contact direction $\epsilon_a = \delta_a/h$, $\delta_a$ is the relative axial penetration in contact/interface/joint axial direction, $h$ is the shear zone thickness, and

- `axial_stiffness` (b) represents the stiffness in the axial direction (local x axis) for 1m penetration.
- `axial_penalty_stiffness` ($E_p$) defines the axial penalty stiffness between upUU and uuUU dof to enforce undrained condition in contact/interface/joint axial direction.
• initial_shear_stiffness \( (E_s) \) is the stiffness in the tangential (shear, local \( y \) or \( z \) axis) directions at 101 kPa axial stress described in Section 104.7.3.3

• axial_viscous_damping is the viscous damping in axial.

• shear_viscous_damping is the viscous damping in shear.

• residual_friction_coefficient \( (\mu_r) \) is the residual frictional parameter as described in Section 104.7.3.3

• shear_zone_thickness \( h \) is the shear zone thickness

• contact_plane_vector Vector defining the normal to the contact/interface/joint plane.

• surface_vector_relative_tolerance defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.92 Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior

The command is:

```
add element # <.> type StressBasedCoupledHardContact_NonLinHardSoftShear
with nodes (<.>, <.>)
axial_stiffness = <Pa>
axial_penalty_stiffness =<Pa>
initial_shear_stiffness = <Pa>
rate_of_softening = <>
size_of_peak_plateau = <>
axial_viscous_damping = <Pa*s>
shear_viscous_damping = <Pa*s>
peak_friction_coefficient_limit = <>
peak_friction_coefficient_rate_of_decrease = <.>
residual_friction_coefficient = <.>
shear_zone_thickness = <m>
contact_plane_vector = (<.>, <.>, <.>);
```

and/or;

```
add element # <.> type StressBasedCoupledHardContact_NonLinHardSoftShear
with nodes (<.>, <.>)
axial_stiffness = <Pa>
axial_penalty_stiffness =<Pa>
initial_shear_stiffness = <Pa>
rate_of_softening = <>
size_of_peak_plateau = <>
axial_viscous_damping = <Pa*s>
shear_viscous_damping = <Pa*s>
peak_friction_coefficient_limit = <>
peak_friction_coefficient_rate_of_decrease = <.>
residual_friction_coefficient = <.>
shear_zone_thickness = <m>
surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

\[
\sigma_a = b \times \epsilon_a
\]
\[
E_a = b
\]  \hspace{1cm} (205.22)

where

$E_a = b$ refers to the axial stiffness in axial contact/interface/joint direction,

$\epsilon_a$ refers to the axial strain in axial contact direction $\epsilon_a = \delta_a/h$,

$\delta_a$ is the relative axial penetration in contact/interface/joint axial direction,

$h$ is the shear zone thickness, and
- **axial_stiffness** (b) represents the stiffness in the axial direction (local x axis) for 1 m penetration.

- **axial_penalty_stiffness** ($E_p$) Defines the axial penalty stiffness between $u_p$ and $u_u$ dof to enforce undrained condition in contact/interface/joint axial direction.

- **initial_shear_stiffness** ($E_s$) Is the stiffness in the tangential (shear, local y or z axis) directions at 101 kPa axial stress, described in Section 104.7.3.4

- **rate_of_softening** ($R_s$) Is the parameter to control the rate of frictional softening described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as

$$\Delta \mu = -\frac{n \ast R_s(\mu_p - \mu_r)}{(\pi/2)^n \theta^{1/n-1}} \ast \cos^2 \theta \Delta \gamma_p$$

(205.23)

$$\theta = \frac{\mu_p - \mu}{\mu_p - \mu_r} (\pi/2)^n$$

(205.24)

where, $R_s$ is the frictional softening rate parameter, $\Delta \gamma_p$ is the plastic shear strain and $n$ represents the size of the peak plateau.

$$\Delta \gamma_p = \sqrt{\Delta \epsilon_{ij}^p \Delta \epsilon_{ij}^p}$$

(205.25)

- **size_of_peak_plateau** ($n$) Is the frictional softening parameter to control the size of plateau as described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as shown in Equation 205.24.

- **axial_viscous_damping** Is the viscous damping in axial.

- **shear_viscous_damping** Is the viscous damping in shear.

- **peak_friction_coefficient_limit** ($\mu_{p0}$) Is the limit to the peak frictional hardening parameter $\mu_p$.

- **peak_friction_coefficient_rate_of_decrease** ($k$) Is the rate of decrease of peak frictional hardening parameter $\mu_p$ with axial stress, described in Section 104.7.3.4

$$\mu_p = \max(\mu_{p0}, \mu_{p0} - k \ast \log(\sigma_a/P_0))$$

(205.26)

where $\mu_{p0}$ is the peak frictional hardening limit, $k$ is the peak frictional parameter rate of decrease and $P_0$ is the reference stress of $P_0 = 101 kPa$. 
• residual_friction_coefficient ($\mu_r$) is the residual frictional parameter as described in Section 104.7.3.4

• shear_zone_thickness $h$ is the shear zone thickness.

• contact_plane_vector Vector defining the normal to the contact/interface/joint plane.

• surface_vector_relative_tolerance defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.93 Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior

The command is:

```plaintext
1 add element # <.> type StressBasedCoupledSoftContact_ElPP1Shear
   with nodes (<.>, <.>)
   initial_axial_stiffness = <Pa>
   stiffening_rate = <>
   max_axial_stiffness = <Pa>
   axial_penalty_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   residual_friction_coefficient = <.>
   shear_zone_thickness = <m>
   contact_plane_vector = (<.>, <.>, <.>);

and/or;

1 add element # <.> type StressBasedCoupledSoftContact_ElPP1Shear
   with nodes (<.>, <.>)
   initial_axial_stiffness = <Pa>
   stiffening_rate = <>
   max_axial_stiffness = <Pa>
   axial_penalty_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   residual_friction_coefficient = <.>
   shear_zone_thickness = <m>
   surface_vector_relative_tolerance = <*>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$\sigma_a = b \ast \exp(a \ast \epsilon_a) \ast \epsilon_a$$

$$E_a = \max(b \ast \exp(a \ast \epsilon_a) \ast (1 + a \ast \epsilon_a), E_{\text{max}})$$

(205.27)

where

$b$ refers to the initial axial stiffness in axial contact/interface/joint direction,

$a$ refers to the stiffening rate in axial contact direction,

$E_{\text{max}}$ refers to the maximum axial stiffness,

$E_a$ refers to the axial stiffness,

$\epsilon_a$ refers to the axial strain in axial contact/interface/joint direction $\epsilon_a = \delta_a/h$,

$\delta_a$ is the relative axial penetration in contact axial direction,

$h$ is the shear zone thickness, and
• initial_axial_stiffness (b) represents the stiffness in the axial direction (local x axis) for 1m penetration.

• stiffening_rate (a) Represents exponential stiffening rate $\exp(sr \ast \epsilon_n)$ in axial direction.

• max_axial_stiffness($E_{\text{max}}$) Defines the maximum stiffness in the axial direction (local x axis) for the contact/interface/joint element.

• axial_penalty_stiffness ($E_p$) Defines the axial penalty stiffness between upU, u and u, u dof to enforce undrained condition in contact/interface/joint axial direction.

• initial_shear_stiffness ($E_s$) Is the stiffness in the tangential (shear, local y or z axis) directions at 101 kPa axial stress described in Section 104.7.3.2

• axial_viscous_damping Is the viscous damping in axial.

• shear_viscous_damping Is the viscous damping in shear.

• residual_friction_coefficient ($\mu_r$) Is the residual friction coefficient described in Section 104.7.3.2

• shear_zone_thickness h Is the shear zone thickness

• contact_plane_vector Vector defining the normal to the contact/interface/joint plane.

• surface_vector_relative_tolerance defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section ??.
### 205.3.4.94 Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior

The command is:

```plaintext
add element # <.> type StressBasedCoupledSoftContact_NonLinHardShear
   with nodes (<.>, <.>)
   initial_axial_stiffness = <Pa>
   stiffening_rate = <>
   max_axial_stiffness = <Pa>
   axial_penalty_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   residual_friction_coefficient = <.>
   axial_zone_thickness = <m>
   contact_plane_vector = (<.>, <.>, <.>);
```

and/or:

```plaintext
add element # <.> type StressBasedCoupledSoftContact_NonLinHardShear
   with nodes (<.>, <.>)
   initial_axial_stiffness = <Pa>
   stiffening_rate = <>
   max_axial_stiffness = <Pa>
   axial_penalty_stiffness = <Pa>
   initial_shear_stiffness = <Pa>
   axial_viscous_damping = <Pa*s>
   shear_viscous_damping = <Pa*s>
   residual_friction_coefficient = <.>
   shear_zone_thickness = <m>
   surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$
\sigma_a = b \cdot \exp(a \cdot \epsilon_a) \cdot \epsilon_a \\
E_a = \max(b \cdot \exp(a \cdot \epsilon_a) \cdot (1 + a \cdot \epsilon_a), E_{\max})
$$

(205.28)

where

- $b$ refers to the initial axial stiffness in axial contact/interface/joint direction,
- $a$ refers to the stiffening rate in axial contact direction,
- $E_{\max}$ refers to the maximum axial stiffness,
- $E_a$ refers to the axial stiffness,
- $\epsilon_a$ refers to the axial strain in axial contact/interface/joint direction $\epsilon_a = \delta_a/h$,
- $\delta_a$ is the relative axial penetration in contact axial direction,
- $h$ is the shear zone thickness, and
• **initial_axial_stiffness** \( (b) \) represents the stiffness in the axial direction (local \( x \) axis) for 1m penetration.

• **stiffening_rate** \( (a) \) Represents exponential stiffening rate \( \exp(sr \cdot \epsilon_n) \) in axial direction.

• **max_axial_stiffness** \( (E_{max}) \) Defines the maximum stiffness in the axial direction (local \( x \) axis) for the contact/interface/joint element.

• **axial_penalty_stiffness** \( (E_p) \) Defines the axial penalty stiffness between up\( U_i \) and \( u_u \) dof to enforce undrained condition in contact/interface/joint axial direction.

• **initial_shear_stiffness** \( (E_s) \) Is the stiffness in the tangential (shear, local \( y \) or \( z \) axis) directions at 101\(^k\)Pa axial stress described in Section 104.7.3.3

• **axial_viscous_damping** Is the viscous damping in axial.

• **shear_viscous_damping** Is the viscous damping in shear.

• **residual_friction_coefficient** \( (\mu_r) \) Is the residual frictional parameter as described in Section 104.7.3.3

• **shear_zone_thickness** \( h \) Is the shear zone thickness

• **contact_plane_vector** Vector defining the normal to the contact/interface/joint plane.

• **surface_vector_relative_tolerance** defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact\_plane\_vector defines a direction from Node I to Node J, that is, **from the first to the second node**. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.95 Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior

The command is:

```plaintext
add element # <> type StressBasedCoupledSoftContact_NonLinHardSoftShear
with nodes (<.>, <.>)
initial_axial_stiffness = <Pa>
stiffening_rate = <>
max_axial_stiffness = <Pa>
axial_penalty_stiffness = <Pa>
initial_shear_stiffness = <Pa>
rate_of_softening = <>
size_of_peak_plateau = <>
axial_viscous_damping = <Pa*s>
shear_viscous_damping = <Pa*s>
peak_friction_coefficient_limit = <>
peak_friction_coefficient_rate_of_decrease = <>
residual_friction_coefficient = <>
shear_zone_thickness = <m>
contact_plane_vector = (<.>, <.>, <.>);
```

and/or;

```plaintext
add element # <> type StressBasedCoupledSoftContact_NonLinHardSoftShear
with nodes (<.>, <.>)
initial_axial_stiffness = <Pa>
stiffening_rate = <>
max_axial_stiffness = <Pa>
axial_penalty_stiffness = <Pa>
initial_shear_stiffness = <Pa>
rate_of_softening = <>
size_of_peak_plateau = <>
axial_viscous_damping = <Pa*s>
shear_viscous_damping = <Pa*s>
peak_friction_coefficient_limit = <>
peak_friction_coefficient_rate_of_decrease = <>
residual_friction_coefficient = <>
shear_zone_thickness = <m>
surface_vector_relative_tolerance = <>;
```

The axial stress \( \sigma_a \) and axial stiffness \( E_a \) in defined as

\[
\sigma_a = b \cdot \exp(a \cdot \epsilon_a) \cdot \epsilon_a
\]

\[
E_a = \max(b \cdot \exp(a \cdot \epsilon_a) \cdot (1 + a \cdot \epsilon_a), E_{\max})
\]

where

- \( b \) refers to the initial axial stiffness in axial contact/interface/joint direction,
- \( a \) refers to the stiffening rate in axial contact direction,
$E_{\text{max}}$ refers to the maximum axial stiffness,
$E_a$ refers to the axial stiffness,
$\epsilon_a$ refers to the axial strain in axial contact/interface/joint direction $\epsilon_a = \delta_a / h$,
$\delta_a$ is the relative axial penetration in contact axial direction,
$h$ is the shear zone thickness, and

- **initial_axial_stiffness** (b) represents the stiffness in the axial direction (local $x$ axis) for $1m$ penetration.
- **stiffening_rate** (a) Represents exponential stiffening rate $exp(st \ast \epsilon_a)$ in axial direction.
- **max_axial_stiffness** ($E_{\text{max}}$) Defines the maximum stiffness in the axial direction (local $x$ axis) for the contact/interface/joint element.
- **axial_penalty_stiffness** ($E_p$) Defines the axial penalty stiffness between upU_U and u_u dof to enforce undrained condition in contact/interface/joint axial direction.
- **initial_shear_stiffness** ($E_s$) Is the stiffness in the tangential (shear, local $y$ or $z$ axis) directions at $101kPa$ axial stress, described in Section 104.7.3.4
- **rate_of_softening** ($R_s$) Is the parameter to control the rate of frictional softening described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as

$$\Delta \mu = -\frac{n \ast R_s(\mu_p - \mu_r)}{(\pi/2)^n \theta^{1/n-1}} \ast \cos^2 \theta \Delta \gamma^p$$  \hspace{1cm} (205.30)

$$\theta = \frac{\mu_p - \mu}{\mu_p - \mu_r} (\pi/2)^n$$  \hspace{1cm} (205.31)

where, $R_s$ is the frictional softening rate parameter, $\Delta \gamma^p$ is the plastic shear strain and $n$ represents the size of the peak plateau.

$$\Delta \gamma^p = \sqrt{\Delta e_p^i \Delta e_p^j}$$  \hspace{1cm} (205.32)

- **size_of_peak_plateau** ($n$) Is the frictional softening parameter to control the size of plateau as described in Section 104.7.3.4. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as shown in Equation 205.31.

- **axial_viscous_damping** Is the viscous damping in axial.
- **shear_viscous_damping** Is the viscous damping in shear.
• **peak_friction_coefficient_limit (μ₀)** is the limit to the peak frictional hardening parameter $μ_p$.

• **peak_friction_coefficient_rate_of_decrease (k)** is the rate of decrease of peak frictional hardening parameter $μ_p$ with axial stress, described in Section 104.7.3.4

$$μ_p = \max(μ_p₀, μ_p₀ - k \log(σ_a/P₀))$$

where $μ_p₀$ is the peak frictional hardening limit, $k$ is the peak frictional parameter rate of decrease and $P₀$ is the reference stress of $P₀ = 101 kPa$.

• **residual_friction_coefficient (μᵣ)** is the residual frictional parameter as described in Section 104.7.3.4

• **shear_zone_thickness h** is the shear zone thickness.

• **contact_plane_vector** Vector defining the normal to the contact/interface/joint plane.

• **surface_vector_relative_tolerance** defines the relative tolerance to find all the contact/interface/joint normals and create multiple contact elements for a given contact node pairs for a conforming surface-to-surface mesh.

**IMPORTANT NOTE No. 1:** contact_plane_vector defines a direction from Node I to Node J, that is, from the first to the second node. If this normal vector is reversed, the contact/interface/joint element behaves as a hook and is likely to create convergence issues.

**IMPORTANT NOTE No. 2:** Two nodes that form the Contact/Interface/Joint Element, need to be placed at the same physical location, coordinates in order to prevent convergence issues when nodes are separated and element tries to close the gap in the very first step.

Description of output by this element can be found in Section 206.8.7.
205.3.4.96 Modeling, Finite Element: Neoprene Isolator Finite Element

(command syntax is in development),

... more on this finite element can be found in Section 102.11 on Page 130 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
Modeling, Finite Element: Lead Core Rubber Isolator/Dissipator Element

(command syntax is in development),

... more on this finite element can be found in Section 102.11 on Page 130 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.98  Modeling, Finite Element: Frictional Pendulum Isolator/Dissipator Finite Element

(command syntax is in development),

... more on this finite element can be found in Section 102.11 on Page 130 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.99 Modeling, Finite Element: Frictional Pendulum Isolator/Dissipator Finite Element
version03

(command syntax is in development),

... more on this finite element can be found in Section 102.11 on Page 130 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
205.3.4.100 Modeling, Damping: Adding Rayleigh Damping

First, define the Rayleigh damping.

```plaintext
add damping # <.> type Rayleigh with a0 = <1/T> a1 = <T> stiffness_to_use = ←
<Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;
```

then apply it to element or node.

```plaintext
add damping # <.> to element # <.>;
add damping # <.> to node # <.>;
```

**NOTE:**

- If the simulation model is a distributed mass system (e.g. using solid brick elements with nonzero density), users should add damping to elements. In other words, if no additional mass were added to nodes, the command add damping to nodes won’t have any effect in ESSI.

- If the simulation model is a lumped mass model (e.g. using the massless beam/truss with lumped mass at nodes), users should add damping to nodes.
205.3.4.101 Modeling, Damping: Adding 3rd Order Caughey Damping

First, define the 3rd-order Caughey damping

```
1  add damping # <.> type Caughey3rd
2  with a0 = <T> a1 = <1/T> a2 = <T^3> stiffness_to_use = ←
   <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;
```

then apply it to element or node.

```
1  add damping # <.> to element # <.>;
1  add damping # <.> to node # <.>;
```

**NOTE:**

- If the simulation model is a distributed mass system (e.g. using solid brick elements with nonzero density), users should add damping to elements. In other words, if no additional mass were added to nodes, the command add damping to nodes won’t have any effect in ESSI.

- If the simulation model is a lumped mass model (e.g. using the massless beam/truss with lumped mass at nodes), users should add damping to nodes.
205.3.4.102 Modeling, Damping: Adding 4th Caughey Damping

First, define the 4th-order Caughey damping

```plaintext
1 add damping #<.> type Caughey4th
2 with a0 = <1/T> a1 = <T> a2 = <T^3> a3 = <T^5> stiffness_to_use = ←
   <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;
```

then apply it to element or node.

```plaintext
1 add damping #<.> to element #<.>;
1 add damping #<.> to node #<.>;
```

NOTE:

- If the simulation model is a distributed mass system (e.g. using solid brick elements with nonzero density), users should add damping to elements. In other words, if no additional mass were added to nodes, the command add damping to nodes won’t have any effect in ESSI.

- If the simulation model is a lumped mass system (e.g. using the massless beam/truss with lumped mass at nodes), users should add damping to nodes.
205.3.4.103 Modeling, Constraints and Supports: Adding Constraints or Supports

```plaintext
fix node # <.> dofs [ux uy uz p Ux Uy Uz rx ry rz all];
```

where at least one of the DOF fixity codes (ux uy uz p Ux Uy Uz rx ry rz all) has to be invoked. These codes are:

- **ux**, translation in \( x \) direction, for structures and solids (solid phase only in \( u - p - U \) and \( u - p \) elements)
- **uy**, translation in \( y \) direction, for structures and solids (solid phase only in \( u - p - U \) and \( u - p \) elements)
- **uz**, translation in \( z \) direction, for structures and solids (solid phase only in \( u - p - U \) and \( u - p \) elements)
- **p**, pore fluid pressure (for fluid phase in \( u - p - U \) and \( u - p \) elements)
- **Ux**, translation of pore fluid phase in \( x \) direction (for \( u - p - U \) elements)
- **Uy**, translation of pore fluid phase in \( y \) direction (for \( u - p - U \) elements)
- **Uz**, translation of pore fluid phase in \( z \) direction (for \( u - p - U \) elements)
- **rx**, rotation around \( x \) axes (for structural elements)
- **ry**, rotation around \( y \) axes (for structural elements)
- **rz**, rotation around \( z \) axes (for structural elements)
- **all**, all applicable DOFs for a given node

Example fix translation \( x \) and \( y \) for node \#3: `fix node # 3 dofs ux uy;`
Example fix all appropriate DOFs for node \#7: `fix node # 7 dofs all;`
205.3.4.104 Modeling, Constraints and Supports: Adding Stochastic Constraints or Supports

Define fixities, boundary conditions for a stochastic node. The command would fix all the polynomial chaos expanded dofs associated with the specified physical dof.

\begin{verbatim}
fix node # <.> stochastic dofs [ux uy uz all];
\end{verbatim}

where at least one of the DOF fixity codes (ux uy uz all) has to be invoked. These codes are

- \texttt{ux}, translation in \textit{x} direction, including all the associated polynomial chaos expanded dofs.
- \texttt{uy}, translation in \textit{y} direction, including all the associated polynomial chaos expanded dofs.
- \texttt{uz}, translation in \textit{z} direction, including all the associated polynomial chaos expanded dofs.
- \texttt{all}, all applicable DOFs for a given node, including all the associated polynomial chaos expanded dofs.

For example,

\begin{verbatim}
fix node # 3 stochastic dofs ux uy;
\end{verbatim}

Fix translation dofs \texttt{ux} and \texttt{uy}, including all the associated polynomial chaos expanded dofs, for stochastic node \# 3.
205.3.4.105  Modeling, Constraints and Supports: Free Constraint or Support

Free the specified DOFs on a designated node.

```plaintext
free node # <.> dofs [ux uy ux p Ux Uy Uz rx ry rz];
```
205.3.4.106 Modeling, Constraints and Supports: Add Tied/Connecte Main-Follower Nodes for the Same DOFs

Add the equal dof for tied/connected nodes for the same degree of freedom.

```
1 add constraint equal_dof with
2 master node # <.> and
3 slave node # <.>
4 dof to constrain <.>;
```
205.3.4.107 Modeling, Constraints and Supports: Adding Tied/Connected, Main-Follower Nodes for Different DOFs

Add the equal dof for tied/connected nodes for different degree of freedom.

```plaintext
1 add constraint equal_dof with node # <.> dof <.> master and node # <.> dof <.> ← slave;
```
205.3.4.108 Modeling, Constraints and Supports: Remove Tied/Connected Main-Follower equal DOFs

Remove the tied/connected nodes equal_dofs.

1 remove constraint equal_dof node # <.>
205.3.4.109  Modeling, Constraints and Supports: Adding Single Point Constraint to Nodes

Define the single point constraint to nodes on a particular degree of freedom for a specified value.

1. add single point constraint to node # <.>
2. dof to constrain <dof_type>
3. constraint value of <.>
205.3.4.110  Modeling, Acceleration Field: Adding Acceleration/Inertia Field

```plaintext
add acceleration field # <.>
ax = <acceleration in x direction>*[L/T^2]
ay = <acceleration in y direction>*[L/T^2]
az = <acceleration in z direction>*[L/T^2];
```

Example adding acceleration induced loading field for (some) elements

```plaintext
add acceleration field # 1
ax = 0*m/s^2
ay = 0*m/s^2
az = -9.81*m/s^2;
```

**NOTE:** see note on page 966 for command

```plaintext
add load # <.> to element # <.> type self_weight use acceleration field # <.>;
```
205.3.4.111  Modeling, Loads: Nodal Loads

The general signature to add loads is

```
1 add load # <.> to node # <.>
2    type <load type> <direction> = <force_amplitude>
3    {more parameters};
```

The load # is a unique number assigned to each load. The node # is the number of a node which has already been defined. The load type refers to the functional form in time or pseudo-time (for static analysis) and can be any of the list

- **linear** Constant rate time dependence.
- **path** Use an arbitrary function defined in an external file.

Each force type except linear have additional parameters which will be explained later.

The force direction refers to the degree of freedom the force will be added to. These force directions are the conjugate in energy of the DOFs defined earlier. These are,

- **Fx**, force in $x$ direction ¹
- **Fy**, force in $y$ direction ¹
- **Fz**, force in $z$ direction ¹
- **F_fluid_x**, force to the pore fluid phase in $x$ direction ²
- **F_fluid_y**, force to the pore fluid phase in $y$ direction ²
- **F_fluid_z**, force to the pore fluid phase in $z$ direction ²
- **Mx**, moment about $x$ axes ³
- **My**, moment about $y$ axes ³
- **Mz**, moment about $z$ axes ³

Example command for adding three linear forces ($f_x = -10 \times kN, f_y = -10 \times kN, f_z = -10 \times kN$) to node # 1:

---

¹ Applies to solid phase only when connected to coupled elements
² Applies to fluid phase when connected to coupled elements. HOWEVER, please note that these are NOT pore fluid pressures, see section 102.12.1.5 on page 142, in Lecture Notes (Jeremić et al., 1989-2021) (Lecture Notes URL).
³ For elements with rotational DOFs, i.e. beams, shells
The force type refers to the functional dependence in time (or pseudo-time) that the force will have. The possible functional forms have been listed before. Listed are additional parameters which define these forces.

1. **linear**
   
   Receives no extra parameters. In this case the magnitude of the force is interpreted as the magnitude of the force after one second of time (or pseudo-time) has passed.

2. **path**
   
   How to add path loads is in the next page.
205.3.4.112  Modeling, Loads: Nodal Path Loads

To add forces which follow a path other than linear, we have the path_series, for equally spaced time series data, and path_time_series for variable spaced time series data.

The commands are:

```
1  add load # <.> to node # <.> type path_series
2    FORCE_TYPE = <force or moment scale factor>
3    time_step = <T>
4    series_file = "STRING";
```

```
1  add load # <.> to node # <.> type path_time_series
2    FORCE_TYPE = <force or moment scale factor>
3    series_file = "STRING";
```

As before, FORCE_TYPE can be Fx, Fy, Fz, Mx, My, Mz, F_fluidx, F_fluidy, F_fluidz.

The format of the series_file is one column of text for the equally spaced case (path_series) and double column, one for time and second one for data values, for (path_time_series).
205.3.4.113 Modeling, Loads: Nodal Loads From Reactions

Loads can be added from reactions.

The command is:

```
1 add load # <.> to node # <.> type from_reactions;
2 add load # <.> to all nodes type from_reactions;
```

The load # is a unique number assigned to each load. The node # is the number of a node which has already been defined. This DSL applies an external load equal to the reaction calculated at that node. It is useful, for stage loading where a constrained dof gets relaxed. The first command add load for a specified node whereas, the second command applies load for all nodes.

For example:

```
1 add load # 3 to node #5 type from_reactions;
```

Adds an external load to node 5 from its reaction force calculated in previous stage.

```
1 add load # 3 to all nodes type from_reactions;
```

Adds an external load to all nodes from their reaction force calculated in previous stage.
205.3.4.114  Modeling, Loads: Selfweight Element Load

```plaintext
add load # <.>
to element # <.>
type self_weight
use acceleration field # <.>;
```

**NOTE:** since the gravity acceleration field is \( g = 9.81 \text{m/s}^2 \), meaning that there is an increment of 9.81 m/s of velocity each second (please note that this defines a rate of increase in velocity), gravity is then applied in 1 second! This is sometimes (most of the time) too harsh numerically! It helps if one defines an acceleration field of say 0.0981 m/s\(^2\) and then apply it in 100 seconds. This is to be done in command add acceleration field ... on page 961.
205.3.4.115  Modeling, Loads: Selfweight Nodal Load

1  add load # <.> to node # <.> type self_weight use acceleration field # <.>;

**NOTE:** For this command to take effect, there should be concentrated mass defined at the node.
205.3.4.116 Modeling, Loads: 8 Node Brick Surface Load with the Constant Pressure

Surface of 8 node brick element with same pressure magnitudes at all nodes:

```
1 add load # <.> to element # <.> type surface at nodes (<.> , <.> , <.> , <.>) ←
   with magnitude <Pa>;
```

Note: This command works for the dry 8NodeBrick element and the coupled 8NodeBrick_upU element. For the coupled upU element, this command applies all surface load on the solid phase, simulating a drained surface loading condition.

A new command for undrained surface loading on upU element will be added soon...
205.3.4.117  Modeling, Loads: 8 Node Brick Surface Load with Variable Pressure

Surface of 8 node brick element with variable pressure magnitudes at all nodes:

```plaintext
1 add load # <.> to element # <.> type surface at nodes (<.> , <.> , <.> , <.>) ←
    with magnitudes ( <Pa> , <Pa> , <Pa> , <Pa> );
```


### 205.3.4.118 Modeling, Loads: 20Node Brick Surface Load with the Constant Pressure

Surface of 20 node brick element with same pressure magnitudes at all nodes:

<table>
<thead>
<tr>
<th>line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>add load # &lt;.&gt; to element # &lt;.&gt; type surface at nodes (&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;) with magnitude &lt;Pa&gt;;</td>
</tr>
</tbody>
</table>
205.3.4.119  Modeling, Loads: 20 Node Brick, Surface Load with Variable Pressure

Surface of 20 node brick element with variable pressure magnitudes at all nodes:

```
add load # <.> to element # <.> type surface at nodes (<.>, <.>, <.>, <.>, (<.>, <.>, <.>, <.>) with magnitudes (<Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>);
```
205.3.4.120 Modeling, Loads: 27 Node Brick Surface Load with the Constant Pressure

Surface of 27 node brick element with same pressure magnitudes at all nodes:

```plaintext
1 add load # <.> to element # <.> type surface at nodes (<.> , <.> , <.> , <.> , ←<.> , <.> , <.> , <.> , <.> , <.>) with magnitude <Pa>;
```


205.3.4.121 Modeling, Loads: 27 Node Brick Surface Load with Variable Pressure

Surface of 27 node brick element with variable pressure magnitudes at all nodes:

```plaintext
add load # <.> to element # <.> type surface at nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) with magnitudes ( <Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>);
```


205.3.4.122  Modeling, Loads: Removing Loads

Loads can be removed using:

```
1 remove load # <.>
```
205.3.4.123 Modeling, Loads: Domain Reduction Method, DRM

```plaintext
1 add load # <.> type domain reduction method hdf5_file = <string>;
2 add load # <.> type domain reduction method hdf5_file = <string> scale_factor = <.>;
```

- `hdf5_file`: HDF5 file with information for the DRM specification. See section.
- `scale_factor`: Factor to linearly scale the motion.

Creating DRM input in HDF5 format. As shown in Fig. (205.5), eight components are required for the DRM input.

![DRM input in HDF5 format](image)

**Figure 205.5**: Components of DRM input in HDF5 format.

The name of the sub-folders must be exactly the same as shown here.

1. **Elements**: element numbers for DRM elements, a single layer of elements used to add the earthquake motion.
2. **DRM Nodes**: Node numbers for DRM elements.
3. **Is Boundary Node**: used to describe whether each of nodes in “DRM Nodes” is a boundary node or an exterior node.
   - If this value is “1”, the corresponding node in “DRM Nodes” is a boundary node.
   - If this value is “0”, the corresponding node in “DRM Nodes” is an exterior node in the DRM element.
4. **Number of Boundary Nodes**: the number of boundary nodes.
5. Number of Exterior Nodes: the number of exterior nodes.

6. Displacements: the displacement components of input earthquake motion on corresponding DRM Nodes. Displacements are a 2D array,
   - column number represents a time-step,
   - rows represents the displacement in one direction.

If every node has 3 degrees of freedom (DOFs, say ux, uy, and uz), the first three rows represent the input displacements on first DRM node in directions of ux, uy, and uz. Then, next three rows represent the input displacements for the next node. So the total row number should be three times of the number of DRM Nodes.

7. Accelerations: same data structure as displacements, above.

8. Time: real time for each time-step in the input earthquake motion.

Python example script to generate the DRM HDF5-based input is given below. Please note that script only generates simplest possible rigid body motion, and that for any realistic motions, those will have to be created using 1C or 3C seismic motions codes, for example, SW4, SynAcc, fk, Hisada, EDT, MS ESSI, or even SHAKE for 1C motions.

```python
# Created by Jose Antonio Abell Mena
# This file reads old-format DRM input files and translates them into new
# HDF5-based format.

# This file produces a rigid body input to the DRM layer. That is, all DRM
# nodes have same X-direction displacement and acceleration. In this case a
# sine wave is used. This is not realistic, its just for demonstration
# purposes. DRM won't work in this case but can be used to verify input if a
# pseudo-static analysis is done (zero density on all elements and apply loads
# with transient analysis.)
# For real input motions, produced using some other means, say SW4, fk,
# Hisada or MS ESSI program, this simple program can be used as an example,
# where DRM input file format used here can be (re)used, while those real
# motions are read form output of above mentioned programs.

import scipy as sp
import h5py
import time

# Write elements and nodes data
elements = sp.loadtxt("DRMelements.txt",dtype=sp.int32)
exterior_nodes = sp.loadtxt("DRMexterior.txt",dtype=sp.int32)
boundary_nodes = sp.loadtxt("DRMbound.txt",dtype=sp.int32)
```
Ne = sp.array(exterior_nodes.size)
Nb = sp.array(boundary_nodes.size)

Nt = Ne+Nb

all_nodes = sp.hstack((boundary_nodes, exterior_nodes))
is_boundary_node = sp.zeros(Nt, dtype=sp.int32)
is_boundary_node[0:Nb] = 1

h5file = h5py.File("small.h5.driminput","w")

h5file.create_dataset("Elements", data=elements)
h5file.create_dataset("DRM Nodes", data=all_nodes)

# This array has 1 if the node at the corresponding position in "DRM nodes" array is a boundary node and zero if not
h5file.create_dataset("Is Boundary Node", data=is_boundary_node)

h5file.create_dataset("Number of Exterior Nodes", data=Ne)
h5file.create_dataset("Number of Boundary Nodes", data=Nb)

# Write timestamp (time format used is that of c "asctime"例: Tue Mar 13 10:17:09 2009)
localtime = time.asctime( time.localtime(time.time()) )
h5file.create_dataset("Created",data=str(localtime))

# Generate motions

t = sp.linspace(0,10,1001)
w = 2*sp.pi/0.5
d = sp.sin(w*t)
a = -w**2*sp.sin(w*t)

# Output accelerations, displacements and time-vector

h5file.create_dataset("Time", data=t)
acc = h5file.create_dataset("Accelerations", (3*Nt,len(t)), dtype=sp.double)
dis = h5file.create_dataset("Displacements", (3*Nt,len(t)), dtype=sp.double)

for node_index in range(Nt):
    acc[3*node_index,:] = a
    acc[3*node_index+1,:] = 0*a #Zero acceleration in y and z
    acc[3*node_index+2,:] = 0*a
    dis[3*node_index,:] = d
    dis[3*node_index+1,:] = 0*d #Zero displacement in y and z
    dis[3*node_index+2,:] = 0*d

h5file.close()
205.3.4.124 Modeling, Wave Field for Creating DRM Loads: Add Wave Field

```
add wave field # <.> with
  acceleration_filename = <string>
  unit_of_acceleration = <L/T^2>
  displacement_filename = <string>
  unit_of_displacement = <L>
  add_compensation_time = <T>
  motion_depth = <L>
  monitoring_location = <within_soil_layer|equivalent_rock_outcropping>
  soil_profile_filename = <string>
  unit_of_Vs = <L/T>
  unit_of_rho = <M/L^3>
  unit_of_damping = <absolute|percent>
  unit_of_thickness = <L>
;
```

Example adding a wave field

```
add wave field # 1 with
  acceleration_filename = "acc.txt"
  unit_of_acceleration = 1 * m/s^2
  displacement_filename = "dis.txt"
  unit_of_displacement = 1 * m
  add_compensation_time = 0.5 * s
  motion_depth = 0 * m
  monitoring_location = within_soil_layer
  soil_profile_filename = "soil_profile.txt"
  unit_of_Vs = 1 * m/s
  unit_of_rho = 1 * kg/m^3
  unit_of_damping = absolute
  unit_of_thickness = 1*m
;
```

where:

- No (or #)<.> is the unique wave field ID. The wave field ID does not have to be sequential, any unique positive integer number can be used. Each wave field is a 1C wave field. The wave field does not have a direction. Later, if users want to add load with the wave field, users should specify the direction with each wave field.

- `acceleration_filename` is the filename of a plain text file, which contains the acceleration of the input motion. The file should have two columns, where the first column is the accumulated time and the second column is the corresponding acceleration. For the DRM loading from a wave field, if the simulation time is longer than the earthquake motion, the remaining simulation will continue with zero motions. The wave field does NOT conduct any base correction on the input
motion, so the simulation results may have permanent deformation after the earthquake. If users want to have base corrections, users should pre-process the earthquake motion by themselves.

- 

    displacement_filename is the filename of a plain text file, which contains the displacement of the input motion. The file should have two columns, where the first column is the accumulated time and the second column is the corresponding displacement. For the DRM loading from a wave field, if the simulation time is longer than the earthquake motion, the remaining simulation will continue with zero motions. The wave field does NOT conduct any base correction on the input motion, so the simulation results may have permanent deformation after the earthquake. If users want to have base corrections, users should pre-process the earthquake motion by themselves.

- add_compensation_time is a feature to add zero-motion in the beginning and at the end of the earthquake motion. Since the wave propagation theory is solving the wave equation in frequency domain (steady state), without additional zeros, the beginning and the end of wave may be mixed up. If the user does not want to add the additional zeros, user can specify 0*s.

- motion_depth is the depth of the input motion. Usually, the motion_depth is at the surface (namely, 0*m). Later, users can specify the request depth for deconvolution. However, users can also specify a specific depth of the input motion. In this case, users can request both convolution and deconvolution.

    - If the request depth is deeper than this input acceleration depth, the wave propagation will generate the deconvolution results.

    - If the request depth is shallower than this input acceleration depth, the wave propagation will generate the convolution results. It is recommended to add a small damping for wave convolution.

The acceleration depth is the relative depth to the soil surface, so both negative and positive depth are acceptable and result in the same results.

- monitoring_location is the location of the earthquake monitoring station. When the monitoring location is within soil layer, the wave propagation is conducted inside the soil layer directly. When the monitoring location is equivalent rock outcropping, wave deconvolution is conducted back to the bedrock first and then propagate into the soil layers.

- soil_profile_filename contains the soil properties for each layer. The soil profile file should have four columns, which are the shear wave velocity, density, damping ratio and thickness of each
layer respectively. The soil layers should be from the soil surface to the bedrock. The last layer is bedrock, which has three columns only. User should NOT give the thickness of the last layer.

One Example of soil profile file is given below.

```
// Vs Vp rho damp thickness
200 333 2000 0.03 35
250 408 2000 0.04 35
2000 3400 2400 0.05
```
205.3.4.125 Modeling, Wave Field for Creating DRM Loads: Deconvolution

This command performs deconvolution or convolution of given motions at surface or certain depth and writes accelerations, velocities and displacements in 3 directions, in files, that can then be used to create DRM loads.

```bash
1 generate wave propagation results of wave field
2   # <.> at depth <L> to <output_filename_prefix>
```

One example of deconvolution is

```bash
1 generate wave propagation results of wave field
2   # 1 at depth -60*m to "Northridge_record" ;
```

where

- wave field # <.> specifies the wave field number which will be used for wave propagation.
- depth <L> is the request depth of the output motion.
  - If the request depth is deeper than the input motion that defined in the wave field, this command will generate the deconvolution results.
  - If the request depth is shallower than the input motion that defined in the wave field, this command will generate the convolution results. It is recommended to add a small damping for wave convolution.

The depth specifies the relative location between the soil surface and the request depth, which means both positive and negative depth are acceptable and will result in the same results.

- output_filename_prefix specified the prefix of the output filenames. This command will generate 3 output files, whose suffix are at_str(depth)_acc.txt, at_str(depth)_vel.txt, at_str(depth)_dis.txt.
205.3.4.126  Modeling, Wave Field for Creating DRM Loads: Deconvolution to DRM

This command performs deconvolution or convolution of given motions at surface or certain depth and directly generates DRM loads in 1, 2, or 3 directions.

```plaintext
1 generate DRM motion file from wave field
2   # <..> in direction <ux|uy|uz>
3   soil_surface at z = <L>
4   hdf5_file = <string> ;
```

One example of deconvolution to DRM is

```plaintext
1 generate DRM motion file from wave field
2   # <..> in direction <ux|uy|uz>
3   soil_surface at z = <L>
4   hdf5_file = <string> ;
```

```plaintext
1 generate DRM motion file from wave field
2   # <..> in direction <ux|uy|uz>
3   # <..> in direction <ux|uy|uz>
4   soil_surface at z = <L>
5   hdf5_file = <string> ;
```

where:

- in direction <ux,uy,uz> specifies the direction of the wave field. Each wave field is a 1C wave field. At most 3 wave fields can be associated with the load.

- soil_surface specifies the relation between the FEM coordinate systems and the soil profile depths inside the wave field. The soil surface should always be above the DRM nodes. Namely, soil surface is generally the surface between the soil and the structure, NOT the bedrock surface.

- hdf5_file specifies the HDF5 file which contain the information about the DRM elements and DRM nodes.
205.3.4.127 Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident SV Wave Potential Magnitude

```
add wave field # <.> type inclined_plane_wave with
  anticlockwise_angle_of_SV_wave_plane_from <x|y|z> = <degrees>
  SV_incident_magnitude = <L^2>
  SV_incident_angle = <degrees>
  SV_incident_frequency = <1/T>
  motion_time_step = <T>
  number_of_time_steps = <.>
  soil_profile_filename = <string>
  soil_surface at <x|y|z> = <L>
  unit_of_vs_and_vp = <L/T>
  unit_of_rho = <M/L^3>
  unit_of_damping = <absolute|percent>
  unit_of_thickness = <L>
;
```

Example of adding an inclined plane wave field

```
add wave field # 1 type inclined_plane_wave with
  anticlockwise_angle_of_SV_wave_plane_from x= 30
  SV_incident_magnitude = 2*m^2
  SV_incident_angle = 60
  SV_incident_frequency = 5/s
  motion_time_step = 0.01*s
  number_of_time_steps = 600
  soil_profile_filename = "soil.txt"
  soil_surface at z = 0*m
  soil_surface at z = 0*m
  unit_of_vs_and_vp = 1*m/s
  unit_of_rho = 1*kg/m^3
  unit_of_damping = absolute
  unit_of_thickness = 1*m;
```

where:

- No (or #)<.> is the unique wave field ID/number. The wave field ID does not have to be sequential, any unique positive integer number can be used. Each wave field is an inclined plane SV wave field.

- `anticlockwise_angle_of_SV_wave_plane_from <x|y|z>` specifies the orientation of the inclined wave field propagation plane. User should give the anticlockwise angle in degrees between the wave propagation plane and the specified reference axis. The reference axis could be x or y or z. As shown in figure 205.7, the `anticlockwise_angle_of_SV_wave_plane_from x axis` is $\alpha$.

- `SV_incident_magnitude` specifies the incident SV wave potential magnitude. The displacement magnitude of incident SV wave is related to the potential magnitude as follows: $|u| = \phi \omega / V_s$. 

where $|u|$ is the displacement magnitude, $\phi$ is potential magnitude, $\omega$ is incident angular frequency and $V_s$ is the shear wave velocity of the incident soil/rock layer.

- $SV_{\text{incident\_angle}}$ specifies the inclination angle of incident SV wave, measured from vertical axis of wave plane to the wave propagation axis. In figure 205.7, the incident angle of SV wave is $\theta$.

- $\text{motion\_time\_step}$ is the time step/interval used for discretizing the harmonic motion into time domain.

- $\text{number\_of\_time\_steps}$ is the number of total time steps for the discretized harmonic motion.

- $\text{soil\_profile\_filename}$ is a file name for a file that contains the soil properties for each layer. The soil profile file should have five columns: (i) shear wave velocity, (ii) compressional wave velocity, (iii) density, (iv) damping ratio and (v) thickness of each layer respectively. The soil layers count from the soil surface to the bedrock. The last layer is bedrock, which has four columns only. User should NOT give the thickness of the last layer, as it extends into halfspace.

One Example of soil profile file is given below.

```plaintext
1 // Vs Vp rho damp thickness
2 200 333 2000 0.02 100
3 250 408 2000 0.02 200
4 2000 3400 2400 0.02
```
soil_surface at <x|y|z> defines the location of soil surface in the global coordinate system of Real-ESSI.
205.3.4.128 Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident SV Wave Time Series Signal

```
add wave field # <.> type inclined_plane_wave with
  anticlockwise_angle_of_SV_wave_plane_from <x|y|z> = <degrees>
  SV_incident_acceleration_filename = <string>
  unit_of_acceleration = <L/T^2>
  SV_incident_displacement_filename = <string>
  unit_of_displacement = <L>
  SV_incident_angle = <degrees>
  add_compensation_time = <T>
  source_location = (<T>, <T>, <T>)
  soil_profile_filename = <string>
  soil_surface at <x|y|z> = <L>
  unit_of_vs_and_vp = <L/T>
  unit_of_rho = <M/L^3>
  unit_of_damping = <absolute|percent>
  unit_of_thickness = <L>
;
```

Example of adding an inclined plane wave field

```
add wave field # 1 type inclined_plane_wave with
  anticlockwise_angle_of_SV_wave_plane_from x = 0
  SV_incident_acceleration_filename = "Kobe_acc.txt"
  unit_of_acceleration = 1*m/s^2
  SV_incident_displacement_filename = "Kobe_disp.txt"
  unit_of_displacement = 1*m
  SV_incident_angle = 15
  add_compensation_time = 0.5*s
  source_location = (-150*m, 0*m, -100*m)
  soil_profile_filename = "soil_profile.txt"
  soil_surface at z = 0*m
  unit_of_vs_and_vp = 1*m/s
  unit_of_rho = 1*kg/m^3
  unit_of_damping = absolute
  unit_of_thickness = 1*m;
```

where:

- No (or #)<.> is the unique wave field ID/number. The wave field ID does not have to be sequential, any unique positive integer number can be used. Each wave field is an inclined plane SV wave field.

- `anticlockwise_angle_of_SV_wave_plane_from <x|y|z>` specifies the orientation of the inclined wave field propagation plane. User should give the anticlockwise angle in degrees between the wave propagation plane and the specified reference axis. The reference axis could be x or y or z. 

z. As shown in figure 205.7, the \( \text{anticlockwise_angle_of_SV_wave_plane_from} \ x \ \text{axis} \) is \( \alpha \).

![Diagram of inclined plane wave field plane location with respect to SSI system orientation.](image)

Figure 205.7: Illustration of inclined plane wave field plane location with respect to SSI system orientation.

- \( \text{SV\_incident\_acceleration\_filename} \) is the filename of a plain text file, which contains the acceleration of the input motion. The file should have two columns, where the first column is the accumulated time and the second column is the corresponding acceleration. For the DRM loading from a wave field, if the simulation time is longer than the earthquake motion, the remaining simulation will continue with zero motions. The wave field implementation in Real-ESSI does NOT conduct any baseline correction on the input motion. The users should perform baseline correction for earthquake motions by themselves.

- \( \text{SV\_incident\_displacement\_filename} \) is the filename of a plain text file, which contains the displacement of the input motion. The file should have two columns, where the first column is the accumulated time and the second column is the corresponding displacement. For the DRM loading from a wave field, if the simulation time is longer than the earthquake motion, the remaining simulation will continue with zero motions. The wave field implementation in Real-ESSI does NOT conduct any baseline correction on the input motion. The users should perform baseline correction for earthquake motions by themselves.

- \( \text{SV\_incident\_angle} \) specifies the inclination angle of incident SV wave, measured from vertical axis of wave plane to the wave propagation axis. In figure 205.7, the incident angle of SV wave is \( \theta \).
• `add_compensation_time` is a feature to add zero-motion in the beginning and at the end of the earthquake motion. If the user does not want to add the additional zeros, user can specify compensation time as 0s.

• `source_location` specify the location of seismic source, where the seismic motion is input as in `SV_incident_acceleration_filename` and `SV_incident_displacement_filename`, it is used for determining phase of the wave, and the source location can be inside or outside of the model.

• `soil_profile_filename` is a file name for a file that contains the soil properties for each layer. The soil profile file should have five columns: (i) shear wave velocity, (ii) compressional wave velocity, (iii) density, (iv) damping ratio and (v) thickness of each layer respectively. The soil layers count from the soil surface to the bedrock. The last layer is bedrock, which has four columns only. User should NOT give the thickness of the last layer, as it extends into half-space.

One Example of soil profile file is given below.

```
1 // Vs Vp rho damp thickness
2 200 333 2000 0.02 100
3 250 408 2000 0.02 200
4 2000 3400 2400 0.02
```

• `soil_surface at <x|y|z>` defines the location of soil surface in the global coordinate system of Real-ESSI.
205.3.4.129 Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident P Wave Potential Magnitude

```
add wave field # <.> type inclined_plane_wave with
  anticlockwise_angle_of_P_wave_plane_from <x|y|z> = <degrees>
P_incident_magnitude = <L^2>
P_incident_angle = <degrees>
P_incident_frequency = <1/T>
motion_time_step = <T>
number_of_time_steps = <.>
soil_profile_filename = <string>
soil_surface at <x|y|z> = <L>
unit_of_vs_and_vp = <L/T>
unit_of_rho = <M/L^3>
unit_of_damping = <absolute|percent>
unit_of_thickness = <L>
;
```

Example adding an inclined plane wave field

```
add wave field # 1 type inclined_plane_wave with
  anticlockwise_angle_of_P_wave_plane_from x= 30
  P_incident_magnitude = 2*m^2
  P_incident_angle = 60
  P_incident_frequency = 5/s
  motion_time_step = 0.01*s
  number_of_time_steps = 600
  soil_profile_filename = "soil.txt"
  soil_surface at z = 0*m
  unit_of_vs_and_vp = 1*m/s
  unit_of_rho = 1*kg/m^3
  unit_of_damping = absolute
  unit_of_thickness = 1*m;
```

where:

- No (or #)<.> is the unique wave field ID/number. The wave field ID does not have to be sequential, any unique positive integer number can be used. Each wave field is an inclined plane P wave field.

- `anticlockwise_angle_of_P_wave_plane_from <x|y|z>` specifies the orientation of the inclined wave propagation plane. User should give the anticlockwise angle in degrees between the wave propagation plane and the specified reference axis. The reference axis could be x or y or z. As shown in figure 205.7, the anticlockwise_angle_of_P_wave_plane_from x axis is \( \alpha \).

- `P_incident_magnitude` specifies the incident P wave potential magnitude. The displacement magnitude of incident P wave is related to the potential magnitude as following: \(|u| = \phi \omega / V_p|\).
where $|u|$ is the displacement magnitude, $\phi$ is potential magnitude, $\omega$ is incident angular frequency and $V_p$ is the compressional wave velocity of the incident layer.

- $P_{\text{incident\_angle}}$ specifies the inclination of incident P wave. The angle is measured from vertical axis of wave plane to the wave propagation axis. In figure 205.7, the incident angle of P wave is $\theta$.

- $\text{motion\_time\_step}$ is the time interval when discretized the harmonic motion into time domain.

- $\text{number\_of\_time\_steps}$ is the number of total time steps for the discretized harmonic motion.

- $\text{soil\_profile\_filename}$ is a file name for a file that contains the soil properties for each layer. The soil profile file should have fives columns: (i) shear wave velocity, (ii) compressional wave velocity, (iii) density, (iv) damping ratio and (v) thickness of each layer respectively. The soil layers count from the soil surface to the bedrock. The last layer is bedrock, which has four columns only. User should NOT give the thickness of the last layer, as it extends into halfspace.

One Example of soil profile file is given below.

```
1 // Vs Vp rho damp thickness
2 200 333 2000 0.02 100
3 250 408 2000 0.02 200
4 2000 3400 2400 0.02
```

- $\text{soil\_surface\ at\ <x|y|z>}$ defines the location of soil surface in the global coordinate system of Real-ESSI.
205.3.4.130 Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident P Wave Time Series Signal

```plaintext
add wave field # <.> type inclined_plane_wave with
  anticlockwise_angle_of_P_wave_plane_from <x|y|z> = <degrees>
P_incident_acceleration_filename = <string>
  unit_of_acceleration = <L/T^2>
P_incident_displacement_filename = <string>
  unit_of_displacement = <L>
P_incident_angle = <degrees>
  add_compensation_time = <T>
source_location = (<T>, <T>, <T>)
soil_profile_filename = <string>
  unit_of_vs_and_vp = <L/T>
soil_surface at <x|y|z> = <L>
unit_of_rho = <M/L^3>
unit_of_damping = <absolute|percent>
unit_of_thickness = <L>
```

Example of adding an inclined plane wave field

```plaintext
add wave field # 1 type inclined_plane_wave with
  anticlockwise_angle_of_P_wave_plane_from x = 0
P_incident_acceleration_filename = "Kobe_acc.txt"
  unit_of_acceleration = 1*m/s^2
P_incident_displacement_filename = "Kobe_disp.txt"
  unit_of_displacement = 1*m
P_incident_angle = 15
  add_compensation_time = 0.5*s
source_location = (-150*m, 0*m, -100*m)
soil_profile_filename = "soil_profile.txt"
  unit_of_vs_and_vp = 1*m/s
soil_surface at z = 0*m
unit_of_rhoe = 1*kg/m^3
unit_of_damping = absolute
unit_of_thickness = 1*m;
```

where:

- No (or #)<.> is the unique wave field ID/number. The wave field ID does not have to be sequential, any unique positive integer number can be used. Each wave field is an inclined plane P wave field.

- `anticlockwise_angle_of_P_wave_plane_from <x|y|z>` specifies the orientation of the inclined wave field propagation plane. User should give the anticlockwise angle in degrees between the wave propagation plane and the specified reference axis. The reference axis could be x or y or
z. As shown in figure 205.7, the \texttt{anticlockwise_angle_of_P_wave_plane_from_x_axis} is $\alpha$.

- \texttt{P_incident_acceleration_filename} is the filename of a plain text file, which contains the acceleration of the input motion. The file should have two columns, where the first column is the accumulated time and the second column is the corresponding acceleration. For the DRM loading from a wave field, if the simulation time is longer than the earthquake motion, the remaining simulation will continue with zero motions. The wave field implementation in Real-ESSI does NOT conduct any baseline correction on the input motion. The users should perform baseline correction for earthquake motions by themselves.

- \texttt{P_incident_displacement_filename} is the filename of a plain text file, which contains the displacement of the input motion. The file should have two columns, where the first column is the accumulated time and the second column is the corresponding displacement. For the DRM loading from a wave field, if the simulation time is longer than the earthquake motion, the remaining simulation will continue with zero motions. The wave field implementation in Real-ESSI does NOT conduct any baseline correction on the input motion. The users should perform baseline correction for earthquake motions by themselves.

- \texttt{P_incident_angle} specifies the inclination angle of incident P wave, measured from vertical axis of wave plane to the wave propagation axis. In figure 205.7, the incident angle of P wave is $\theta$.

- \texttt{add_compensation_time} is a feature to add zero-motion in the beginning and at the end of the earthquake motion. If the user does not want to add the additional zeros, user can specify compensation time as 0*s.

- \texttt{source_location} specify the location of seismic source, where the seismic motion is input as in \texttt{P_incident_acceleration_filename} and \texttt{P_incident_displacement_filename}, it is used for determining phase of the wave, and the source location can be inside or outside of the model.

- \texttt{soil_profile_filename} is a file name for a file that contains the soil properties for each layer. The soil profile file should have fives columns: (i) shear wave velocity, (ii) compressional wave velocity, (iii) density, (iv) damping ratio and (v) thickness of each layer respectively. The soil layers count from the soil surface to the bedrock. The last layer is bedrock, which has four columns only. User should NOT give the thickness of the last layer, as it extends into half-space.

One Example of soil profile file is given below.

```
1 # Vs Vp rho damp thickness
2 200 333 2000 0.02 100
3 250 408 2000 0.02 200
```
- `soil_surface at <x|y|z>` defines the location of soil surface in the global coordinate system of Real-ESSI.
205.3.4.131 Modeling, Wave Field for Creating DRM Loads: DRM Inclined Motion

This command generates inclined DRM motion with pre-defined inclined wave field.

```plaintext
generate DRM motion file from wave field # <.> hdf5_file = <string>;
```

One example of generating inclined DRM motion is:

```plaintext
generate DRM motion file from wave field # 1 hdf5_file = "DRMinput.hdf5";
```

where

- `wave field # <.>` specifies the inclined plane wave field number which will be used for wave propagation.

- `hdf5_file` specifies the HDF5 file which contains the geometric information about the DRM elements and DRM nodes.
205.3.4.132 Modeling, Imposed Motions: through Loads, Motion Time History, Constant Time Step

Impose motions (displacements, velocities and accelerations) through loads. This one is used if time increment is constant during the analysis. Input files have one column only, corresponding file for displacements, velocities, and accelerations.

```
add load # <.> type imposed motion to node # <.> dof DOFTYPE
time_step = <T>
displacement_scale_unit = <L>
displacement_file = "filename"
velocity_scale_unit = <L/T>
velocity_file = "filename"
acceleration_scale_unit = <L/L^2>
acceleration_file = "filename";
```

The above command generates load to the corresponding node to get the applied imposed motion.
205.3.4.133 Modeling, Imposed Motions: through Loads, Stochastic Motion Time History, Constant Time Step

Impose stochastic motions (uncertain displacements, velocities and accelerations) through stochastic loads. This one is used if time increment is constant during the analysis.

```plaintext
add load # <.> type imposed random motions to node # <.> dof DOFTYPE
1
time_step = <T>
2
displacement_scale_unit = <L>
3
displacement_file = "filename"
4
velocity_scale_unit = <L/T>
5
velocity_file = "filename"
6
acceleration_scale_unit = <L/L^2>
7
acceleration_file = "filename"
8
penalty_stiffness = <N/L>
9
using double product # <.>;
```

where

- **DOFTYPE** specify the dof to impose the uncertain motion. It can be either $u_x$, $u_y$ or $u_z$.

- **time_step** specify the time step of the imposed uncertain motion.

- **displacement_scale_unit** specify the scale unit of the imposed uncertain displacement polynomial chaos (PC) coefficients.

- **displacement_file** specify the filename of a text file containing PC coefficients of uncertain displacement random process. The number of rows of the file content should be equal to total number of polynomial chaos basis of the displacement random process. The number of columns of the file content should be equal to total number of time steps of the displacement random process. The value at the $i^{th}$ row and $j^{th}$ column of the file gives the PC coefficient of the $i^{th}$ PC basis of the displacement random process at the $j^{th}$ time step.

- **velocity_scale_unit** specify the scale unit of the imposed uncertain velocity polynomial chaos (PC) coefficients.

- **velocity_file** specify the filename of a text file containing PC coefficients of uncertain velocity random process. The number of rows of the file content should be equal to total number of polynomial chaos basis of the velocity random process. The number of columns of the file content should be equal to total number of time steps of the velocity random process. The value at the $i^{th}$ row and $j^{th}$ column of the file gives the PC coefficient of the $i^{th}$ PC basis of the velocity random process at the $j^{th}$ time step.
• `acceleration_scale_unit` specify the scale unit of the imposed uncertain acceleration polynomial chaos (PC) coefficients.

• `acceleration_file` specify the filename of a text file containing PC coefficients of uncertain acceleration random process. The number of rows of the file content should be equal to total number of polynomial chaos basis of the acceleration random process. The number of columns of the file content should be equal to total number of time steps of the acceleration random process. The value at the $i^{th}$ row and $j^{th}$ column of the file gives the PC coefficient of the $i^{th}$ PC basis of the acceleration random process at the $j^{th}$ time step.

• `penalty_stiffness` specify the penalty stiffness for the input of uncertain motion using penalty method. The penalty stiffness is expected to be several magnitudes, e.g., $10^3 \sim 10^6$, larger than the elemental stiffness.

• `double_product #` specify the ID of the double product that would be used in the formation of stochastic force. In stochastic finite element method (FEM), the first PC basis for this double product should come from the PC representation of uncertain FEM system response, e.g., uncertain structural displacement. The second PC basis for this double product should come from the uncertain imposed motion representation.
205.3.4.134 Modeling, Imposed Motions: through Loads, Stochastic Random Process Motions, Constant Time Step

Impose a defined random process motions.

This one is used if time increment is constant during the analysis.

```plaintext
add load # <.> type imposed random motions to node # <.> dof DOFTYPE
1
time_step = <T>
2
uncertain_displacement = random field # <.>
3
displacement_scale_unit = <L>
4
penalty_stiffness = <N/L>
5
using double product # <.>;
6
```

where

- **DOFTYPE** specify the dof to impose the uncertain motion. It can be either ux, uy or uz.
- **time_step** specify the time step of the imposed uncertain motion.
- **uncertain_displacement** specify the imposed uncertain motion through a defined random field/process.
- **displacement_scale_unit** specify the scale unit of the imposed uncertain displacement.
- **penalty_stiffness** specify the penalty stiffness for the input of uncertain motion using penalty method. The penalty stiffness is expected to be several magnitudes, e.g., $10^3 \sim 10^6$, larger than the elemental stiffness.
- **double product #** specify the ID of the double product that would be used in the formation of stochastic force. In stochastic finite element method (FEM), the first PC basis for this double product should come from the PC representation of uncertain FEM system response, e.g., uncertain structural displacement. The second PC basis for this double product should come from the uncertain imposed motion representation.
205.3.4.135 Modeling, Imposed Motions: through Loads, Motion Time History, Variable Time Step

Impose motions (displacements, velocities and accelerations) through loads. This one is used if time increment is variable during the analysis. Input files have two columns, first column is time and the second column in corresponding file for displacements, velocities, and accelerations. Time steps have to be the same in each file.

```
1 add load # <.> type imposed motion to node # <.> dof DOFTYPE
2 displacement_scale_unit = <displacement>
3 displacement_file = "filename"
4 velocity_scale_unit = <velocity>
5 velocity_file = "filename"
6 acceleration_scale_unit = <acceleration>
7 acceleration_file = "filename";
```

The above command generates load to the corresponding node to get the applied imposed motion.
205.3.4.136 Modeling, Imposed Motions: Adding Load for Uniform Acceleration Time History

Defines a non-inertial reference frame from which all displacements are measured. This reference frame (fixed to the base of the model) accelerates according to a given acceleration record. All output quantities are derived from this relative coordinate system (not-inertial). To get total displacements, the twice-integrated acceleration record must be added to the results.

The command is:

```plaintext
1 add load # <.> type uniform acceleration to all nodes dof <.>
2 time_step = <T>
3 scale_factor = <L/T^2>
4 initial_velocity = <L/T>
5 acceleration_file = <string>;
```

Where

- `time_step` is the time step of the record in time units.
- `scale_factor` is a dimensionless factor with which the record is scaled before it’s applied.
- `initial_velocity` initial velocity for all translational DOFs of the system.
- `acceleration_file` string containing the path (relative or absolute) to the record text file.

File format is a single value of the record in acceleration units (m/s/s) per line for each time step. If a time-step different from the record is used for analysis, then the record is interpolated linearly.
205.3.4.137 Modeling, Imposed Motions: Remove Imposed Motions

Motions can be removed using:

```
1 remove imposed motion # <.>
```
205.3.4.138 Modeling, Random Variable: Adding Gaussian Random Variables

Gaussian random variable can be added for probabilistic analysis.

The command is:

```
1 add random variable # <.> with Gaussian distribution mean = <.> ←
   standard_deviation = <.>;
```

where:

- `mean` is the mean of the Gaussian random variable
- `standard_deviation` is the standard deviation of the Gaussian random variable

For example:

```
1 add random variable # 1 with Gaussian distribution mean = 3.0 ←
   standard_deviation = 1.0;
```

 Adds a Gaussian random variable 1 with mean equal to 3.0 and standard deviation equal to 1.0.
205.3.4.139 Modeling, Random Variable: Adding Gaussian Random Variables with Location

Gaussian random variable with spatial location can be added for probabilistic analysis. The location information of the defined random variable is important to calculate the correlation structure of random field, that can consist of many random variables.

The command is:

```
1 add random variable # <.> with Gaussian distribution mean = <.> ←
    standard_deviation = <.> at (<L>, <L>, <L>);
```

where:

- **mean** is the mean of the Gaussian random variable
- **standard_deviation** is the standard deviation of the Gaussian random variable

For example:

```
1 add random variable # 1 with Gaussian distribution mean = 3.0 ←
    standard_deviation = 1.0 at (3*m, 0*m, 0*m);
```

Adds a Gaussian random variable 1 with mean equal to 3.0 and standard deviation equal to 1.0 at location \( x = 3 \text{ m}, \ y = 0 \text{ m} \) and \( z = 0 \text{ m} \).
205.3.4.140 Modeling, Random Variable: Adding Lognormal Random Variables

Lognormal random variable can be added for probabilistic analysis.

The command is:

```plaintext
1 add random variable # <.> with Lognormal distribution mean = <.> ←
   standard_deviation = <.>;
```

where:

- `mean` is the mean of the lognormal random variable
- `standard_deviation` is the standard deviation of the lognormal random variable

For example:

```plaintext
1 add random variable # 1 with Lognormal distribution mean = 3.0 ←
   standard_deviation = 1.0;
```

Adds a Lognormal random variable 1 with mean equal to 3.0 and standard deviation equal to 1.0.
205.3.4.141 Modeling, Random Variable: Adding Lognormal Random Variables with Location

Lognormal random variable with spatial location can be added for probabilistic analysis. The location information of the defined random variable is important to calculate the correlation structure of random field, that can consist of many random variables.

The command is:

```
add random variable # <.> with Lognormal distribution mean = <.> ←
standard_deviation = <.> at (<L>, <L>, <L>);
```

where:

- `mean` is the mean of the lognormal random variable
- `standard_deviation` is the standard deviation of the lognormal random variable

For example:

```
add random variable # 1 with Lognormal distribution mean = 3.0 ←
standard_deviation = 1.0 at (3*m, 0*m, 0*m);
```

Adds a Lognormal random variable 1 with mean equal to 3.0 and standard deviation equal to 1.0 at location \( x = 3 \) m, \( y = 0 \) m and \( z = 0 \) m.
205.3.4.142 Modeling, Random Variable: Adding Lognormal Random Variables using Logarithmic Input

Lognormal random variable can be added for probabilistic analysis.

The command is:

```plaintext
1 add random variable # <.> with Lognormal distribution lognormal_mean = <.> ←
   lognormal_standard_deviation = <.>;
```

where:

- `lognormal_mean`: $\mu$ is the mean of the natural logarithm of the lognormal random variable $X$

- `lognormal_standard_deviation`: $\sigma$ is the standard deviation of the natural logarithm of the lognormal random variable $X$

In other words, for lognormal distributed random variable $X$ with parameters $\mu$ and $\sigma$, we have:

$$\ln(X) \sim N(\mu, \sigma)$$

(205.34)

It is noted that the mean $m$ and variance $v$ of lognormal random variable $X$ is related to parameters $\mu$ and $\sigma$ as follows:

$$m = e^{\mu + \sigma^2/2}$$

(205.35)

$$v = e^{2\mu + \sigma^2}(e^{\sigma^2} - 1)$$

(205.36)

For example:

```plaintext
1 add random variable # 1 with Lognormal distribution lognormal_mean = 3.0 ←
   lognormal_standard_deviation = 1.0;
```

adds a Lognormal random variable 1. The natural logarithm of such random variable follows Gaussian distribution with mean equal to 3.0 and standard deviation equal to 1.0.
205.3.4.143  Modeling, Random Variable: Adding Lognormal Random Variables using Logarithmic Input with Location

Lognormal random variable with spatial location can be added for probabilistic analysis. The location information of the defined random variable is important to calculate the correlation structure of random field, which can consist of many random variables.

The command is:

```
add random variable # <.> with Lognormal distribution lognormal_mean = <.> ←
lognormal_standard_deviation = <.> at (<L>, <L>, <L>);
```

where:

- `lognormal_mean`: $\mu$ is the mean of the natural logarithm of the lognormal random variable $X$
- `lognormal_standard_deviation`: $\sigma$ is the standard deviation of the natural logarithm of the lognormal random variable $X$

In other words, for lognormal distributed random variable $X$ with parameters $\mu$ and $\sigma$, we have:

$$\ln(X) \sim N(\mu, \sigma)$$ (205.37)

It is noted that the mean $m$ and variance $v$ of lognormal random variable $X$ is related to parameters $\mu$ and $\sigma$ as follows:

$$m = e^{\mu+\sigma^2/2}$$ (205.38)

$$v = e^{2\mu+\sigma^2}(e^{\sigma^2} - 1)$$ (205.39)

For example:

```
add random variable # 1 with Lognormal distribution lognormal_mean = 3.0 ←
lognormal_standard_deviation = 1.0 at (3*m, 0*m, 0*m);
```

Adds a Lognormal random variable 1 at location $x = 3$m, $y = 0$m and $z = 0$m. The natural logarithm of such random variable follows Gaussian distribution with mean equal to 3.0 and standard deviation equal to 1.0.
205.3.4.144 Modeling, Random Variable: Adding Gamma Random Variables using Shape and Scale Parameters

Random variable following Gamma distribution can be added for probabilistic analysis.

The command is:

```plaintext
1 add random variable # <.> with Gamma distribution shape_parameter = <.> ←
  scale_parameter = <.>;
```

where:

- `shape_parameter` is the shape parameter of the Gamma random variable
- `scale_parameter` is the scale parameter of the Gamma random variable

For example:

```plaintext
1 add random variable # 1 with Gamma distribution shape_parameter = 5.0 ←
  scale_parameter = 2.0;
```

Adds a Gamma random variable 1 with the shape parameter equal to 5.0 and scale parameter equal to 2.0.
205.3.4.145  Modeling, Random Variable: Adding Gamma Random Variables using Shape and Scale Parameters with Location

Random variable following Gamma distribution can be added for probabilistic analysis.

The location information of the defined random variable is important to calculate the correlation structure of random field, that can consist of many random variables.

The command is:

```
add random variable # <.> with Gamma distribution shape_parameter = <.> ← scale_parameter = <.> at (<L>, <L>, <L>);
```

where:

- shape_parameter is the shape parameter of the Gamma random variable
- scale_parameter is the scale parameter of the Gamma random variable

For example:

```
add random variable # 1 with Gamma distribution shape_parameter = 5.0 ← scale_parameter = 2.0 at (3*m, 0*m, 0*m);
```

Adds a Gamma random variable 1 with the shape parameter equal to 5.0 and scale parameter equal to 2.0 at location $x = 3\text{ m}$, $y = 0\text{ m}$ and $z = 0\text{ m}$.
205.3.4.146  Modeling, Random Variable: Adding Gamma Random Variables using Mean and Standard Deviation Parameters

Random variable for given mean and standard deviation parameter following Gamma distribution can be added for probabilistic analysis.

The command is:

```
1 add random variable # <.> with Gamma distribution mean = <.> standard_deviation ← = <.>;
```

where:

- `mean` is the mean of the Gamma random variable
- `standard_deviation` is the standard deviation of the Gamma random variable

For example:

```
1 add random variable # 1 with Gamma distribution mean = 5.0 standard_deviation = ← 2.0;
```

Adds a Gamma random variable 1 with mean equal to 5.0 and standard deviation equal to 2.0.
205.3.4.147 Modeling, Random Variable: Adding Gamma Random Variables using Mean and Standard Deviation Parameters with Location

Random variable for given mean and standard deviation parameter following Gamma distribution can be added for probabilistic analysis.

The location information of the defined random variable is important to calculate the correlation structure of random field, that can consist of many random variables.

The command is:

```plaintext
1 add random variable # <.> with Gamma distribution mean = <.> standard_deviation ← = <.> at (<L>, <L>, <L>);
```

where:

- `mean` is the mean of the Gamma random variable
- `standard_deviation` is the standard deviation of the Gamma random variable

For example:

```plaintext
1 add random variable # 1 with Gamma distribution mean = 5.0 standard_deviation ← 2.0 at (3*m, 0*m, 0*m);
```

Adds a Gamma random variable 1 with mean equal to 5.0 and standard deviation equal to 2.0 at location \(x = 3\, \text{m}, y = 0\, \text{m and } z = 0\, \text{m}.\)
Random variable following Weibull distribution can be added for probabilistic analysis.

The command is:

```plaintext
add random variable # <.> with Weibull distribution shape_parameter = <.> ←
scale_parameter = <.>
```

where:

- `shape_parameter` is the shape parameter of the Weibull random variable
- `scale_parameter` is the scale parameter of the Weibull random variable

For example:

```plaintext
add random variable # 1 with Weibull distribution shape_parameter = 5.0 ←
scale_parameter = 2.0;
```

Adds a Weibull random variable 1 with the shape parameter equal to 5.0 and scale parameter equal to 2.0.
205.3.4.149  Modeling, Random Variable: Adding Weibull Random Variables using Shape and Scale Parameters with Location

Random variable following Weibull distribution can be added for probabilistic analysis.

The location information of the defined random variable is important to calculate the correlation structure of random field, which can consist of many random variables.

The command is:

```plaintext
add random variable # <.> with Weibull distribution shape_parameter = <.> ←
  scale_parameter = <.> at (<L>, <L>, <L>);
```

where:

- **shape_parameter** is the shape parameter of the Weibull random variable
- **scale_parameter** is the scale parameter of the Weibull random variable

For example:

```plaintext
add random variable # 1 with Weibull distribution shape_parameter = 5.0 ←
  scale_parameter = 2.0 at (3*m, 0*m, 0*m);
```

adds a Weibull random variable 1 with the shape parameter equal to 5.0 and scale parameter equal to 2.0 at location \(x = 3\text{m}, y = 0\text{m}\) and \(z = 0\text{m}\).
205.3.4.150 Modeling, Random Variable: Remove Random Variables

Remove random variables.

The command is:

```plaintext
remove random variable # <.> ;
```

For example:

```plaintext
remove random variable # 2;
```

Remove random variable 2 from the analysis.
205.3.4.151 Modeling, Random Variable: Hermite Polynomial Chaos Expansion

Hermite polynomial chaos expansion Xiu (2010) can be performed for random variable with any type of distribution.

The command is:

```
Hermite polynomial chaos expansion to random variable # <.> with order <.>;
```

where:

- order specifies the order of Hermite polynomial chaos expansion

For example:

```
Hermite polynomial chaos expansion to random variable # 1 with order 6;
```

Performs Hermite polynomial chaos expansion to random variable 1 using Hermite polynomial chaos up to order 6.
205.3.4.152 Modeling, Random Variable: Output Hermite Polynomial Chaos Expansion Result

A HDF5 (.hdf5) file contains computed polynomial chaos coefficients of Hermite polynomial chaos expansion for random variable can be generated.

The command is:

```plaintext
generate Hermite polynomial chaos expansion file from random variable # <.> ←
hdf5_file = "file_name";
```

where:

- `file_name` is a string that specifies the name of the output hdf5 file.

The generated hdf5 file contains two datasets:

- Dataset **PC** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis. \( PC_{ij} \) denotes the order of polynomial chaos dimension \( \xi_j \) that contributes to the \( i^{th} \) multidimensional Hermite PC basis.

- Dataset **PC Coefficients** is a column vector. The \( i^{th} \) component of **PC Coefficients** is the polynomial chaos coefficient corresponding to the \( i^{th} \) PC base as described by **PC**.

For example:

```plaintext
generate Hermite polynomial chaos expansion file from random variable # 2 ←
hdf5_file = "PC_RV1.hdf5";
```

Generate HDF5 file named “PC_RV2.hdf5” that contains the computed polynomial chaos coefficients of Hermite polynomial chaos expansion of random variable 2.
205.3.4.153 Modeling, Random Variable: Hermite Polynomial Chaos Expansion & Output Results

Hermite polynomial chaos expansion Xiu (2010) can be performed for random variable with any type of distribution. A HDF5 (.hdf5) file contains computed polynomial chaos coefficients of Hermite polynomial chaos expansion for random variable can be generated.

The command is:

```
1 generate Hermite polynomial chaos expansion file from random variable # <.> ←
   with order <.> hdf5_file = "file_name";
```

where:

- **order** specifies the order of Hermite polynomial chaos expansion.
- **file_name** is a string that specifies the name of the output hdf5 file.

The generated hdf5 file contains two datasets:

- **Dataset PC** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis. $PC_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ multidimensional Hermite PC basis.
- **Dataset PC Coefficients** is a column vector. The $i^{th}$ component of PC Coefficients is the polynomial chaos coefficient corresponding to the $i^{th}$ PC base as described by PC.

For example:

```
1 generate Hermite polynomial chaos expansion file from random variable # 1 with ←
   order 6 hdf5_file = "PC_RV1.hdf5";
```

Perform Hermite polynomial chaos expansion to random variable 1 using Hermite polynomial chaos up to order 6 and generate HDF5 file named “PC_RV1.hdf5” that contains the computed polynomial chaos coefficients.
205.3.4.154 Modeling, Random Field: Adding Random Field with Dimension and Order

Random field with specific Hermite polynomial chaos dimension and order can be added for probabilistic analysis.

The command is:

```
1 add random field # <.> with Hermite polynomial chaos dimension <.> order <.>
```

where:

- **dimension** defines the dimension of Hermite polynomial chaos expansion of the random field
- **order** defines the order of Hermite polynomial chaos expansion of the random field

For example:

```
1 add random field # 1 with Hermite polynomial chaos dimension 4 order 3;
```

adds random field 1 with Hermite polynomial chaos expansion of dimension 4 and order 3.
205.3.4.155 Modeling, Random Field: Define Global Dimension Index of Random Field

Define the global dimension index of local Hermite polynomial chaos (PC) dimension for the uncertainty characterization of the random field.

If the global dimension index is not specified, by default Real-ESSI takes the ID of local Hermite polynomial chaos (PC) dimension as the global dimension index.

Please note that correctly specifying global dimension index for local Hermite PC dimensions of the random field is very important, especially when there are multiple random fields exist in the system and need to compute the triple products of Hermite PC basis of these random fields.

The command is:

```
1 define random field # <> Hermite polynomial chaos dimension # <> as global ← dimension # <>;
```

where:

- **Hermite polynomial chaos dimension** defines the local dimension ID for Hermite polynomial chaos (PC) basis of the random field. It should be an integer no more than the total number of dimensions adopted in the Hermite polynomial chaos (PC) Karhunen Loève expansion of the random field.

- **global dimension** defines the corresponding global dimension index for the local Hermite PC dimension

For example:

```
1 define random field # 1 Hermite polynomial chaos dimension # 1 as global ← dimension # 10;
```

defines the global dimension index of local Hermite PC dimension 1 of random field 1 is 10.
205.3.4.156 Modeling, Random Field: Define Global Dimension Index of Random Field from File Input

Define the global dimension index of local Hermite polynomial chaos (PC) dimension for the uncertainty characterization of the random field using inputs from a text file.

If the global dimension index is not specified, by default Real-ESSI takes the ID of local Hermite polynomial chaos (PC) dimension as the global dimension index.

Please note that correctly specifying global dimension index for local Hermite PC dimensions of the random field is very important, especially when there are multiple random fields exist in the system and need to compute the triple products of Hermite PC basis of these random fields.

The command is:

```
1 define random field # <.> Hermite polynomial chaos dimension from dimension_file = "file_name";
```

where:

- `dimension_file` specifies the name of a text file that contains two columns: The first column is local dimension ID of Hermite PC basis; The second column is corresponding global dimension ID for the local dimension of Hermite PC basis. Comments lines starts with “//”

For example:

```
1 define random field # 1 Hermite polynomial chaos dimension from dimension_file = "dimension_info_RF1.txt";
```

defines the global dimension index of local Hermite PC dimensions for random field 1 with input from a text file “dimension_info_RF1.txt”.

An example file of “dimension_info_RF1.txt” is provided below:

```
//===================================================================
// This file specify the global dimension index of local dimensions of Hermite PC basis
// File should have two columns separated by spaces:
// The first column is local KL dimension ID
// The second column is the global KL dimension ID
//===================================================================
1 10
2 11
3 12
4 13
```
205.3.4.157 Modeling, Random Field: Set Number of Polynomial Chaos Terms of Random Field

Specify the number of polynomial chaos terms of a random field involved in the stochastic finite element analysis. By default, the full polynomial chaos basis of a random field would be used for uncertainty propagation. The specified number of polynomial chaos terms in this command is used for truncation of polynomial chaos basis.

The command is:

\begin{verbatim}
set random field # <.> polynomial_chaos_terms = <.>;
\end{verbatim}

where:

- `polynomial_chaos_terms` defines the number of truncated Hermite polynomial chaos basis of the random field

For example:

\begin{verbatim}
set random field # 1 polynomial_chaos_terms = 100;
\end{verbatim}

Set the number of truncated Hermite polynomial chaos basis of random field 1 to be 100.
205.3.4.158 Modeling, Random Field: Adding Random Field with Zero Correlation

Random field with uncorrelated random variables can be added for probabilistic analysis.

The command is:

```plaintext
add random field # <.> with zero correlation;
```
205.3.4.159 Modeling, Random Field: Adding Random Field with Exponential Correlation

Random field with exponential correlation can be added for probabilistic analysis.

The command is:

```
add random field # <.> with exponential correlation correlation_length = <L> ;
```

where:

- `correlation_length` \( l_c \) defines the correlation length of random field such that the correlation \( \rho(RV_i, RV_j) \) of any two random variables \( RV_i \) and \( RV_j \) is given as:

\[
\rho(RV_i, RV_j) = \exp(-d/l_c) \tag{205.40}
\]

Variable \( d \) is the Euclidean distance between \( RV_i \) and \( RV_j \), that is calculated from the spatial locations of random variables within the random field.

For example:

```
add random field # 1 with exponential correlation correlation_length = 10*m;
```

adds an exponentially correlated random field number 1 with correlation length 10m.
205.3.4.160 Modeling, Random Field: Adding Random Field with Triangular Correlation

Random field with triangular correlation can be added for probabilistic analysis.

The command is:

```
add random field # <.> with triangular correlation correlation_length = <L> ;
```

where:

- `correlation_length` defines the correlation length of random field such that the correlation
  $\rho(RV_i, RV_j)$ of any two random variables $RV_i$ and $RV_j$ is given as:

$$
\rho(RV_i, RV_j) = \max\{1 - d/l_c, 0\}
$$

Variable $d$ is the Euclidean distance between $RV_i$ and $RV_j$, that is calculated from the spatial
locations of random variables within the random field.

For example:

```
add random field # 1 with triangular correlation correlation_length = 10*m;
```

Adds an triangular correlated random field number 1 with correlation length 10m.
205.3.4.161  Modeling, Random Field: Adding Random Field with Exponentially Damped Cosine Correlation

Random field with exponentially damped cosine correlation can be added for probabilistic analysis.

The command is:

```
1 add random field # <.> with exponentially damped cosine correlation ←
  correlation_length = <L> ;
```

where:

- `correlation_length l_c` defines the correlation length of random field such that the correlation $\rho(RV_i, RV_j)$ of any two random variables $RV_i$ and $RV_j$ is given as:

$$
\rho(RV_i, RV_j) = \exp(-d/l_c) \ast \cos(d/l_c)
$$

Variable $d$ is the Euclidean distance between $RV_i$ and $RV_j$, that is calculated from the spatial locations of random variables within the random field.

For example:

```
1 add random field # 1 with exponentially damped cosine correlation ←
  correlation_length = 10*m;
```

adds an exponentially damped cosine correlated random field number 1 with correlation length 10m.
205.3.4.162 Modeling, Random Field: Remove Random Fields

Remove random Fields.

The command is:

```
1 remove random field # <.> ;
```

For example:

```
1 remove random field # 2;
```

Remove random field 2 from the analysis.
205.3.4.163 Modeling, Random Field: Adding Random Variable to Random Field

Add random variable to random field.

The command is:

1 \texttt{add random variable \# <.> to random field \# <.>;} \hspace{1cm}

For example:

1 \texttt{add random variable \# 2 to random field \# 1;} \hspace{1cm}

Adds random variable 2 to random field 1.
205.3.4.164 Modeling, Random Field: Remove Random Variable From Random Field

Remove random variable from random field.

The command is:

```
remove random variable # <.> from random field # <.>;
```

For example:

```
remove random variable # 2 from random field # 1;
```

Remove random variable 2 from random field 1.
205.3.4.165 Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion

Perform Hermite polynomial chaos Karhunen Loève expansion for random field of any arbitrary marginal distribution and correlation structure according to Sakamoto and Ghanem (2002).

The command is:

```
Hermite polynomial chaos Karhunen Loeve expansion to random field # <.> with ←
Hermite polynomial chaos dimension <.> order <.>;
```

Where:

- **dimension**: specifies the number of dimensions of Hermite polynomial chaos to capture the correlation structure of the random field
- **order**: specifies the order of Hermite polynomial chaos to capture the marginal distribution of the random field

For example:

```
Hermite polynomial chaos Karhunen Loeve expansion to random field # 1 with ←
Hermite polynomial chaos dimension 4 order 3;
```

Perform the Hermite polynomial chaos Karhunen Loève expansion for random field 1 using Hermite polynomial chaos of dimension 4 and order 3.
205.3.4.166 Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion with Inverse Order

Perform Hermite polynomial chaos Karhunen Loève expansion for random field of any arbitrary marginal distribution and correlation structure according to Sakamoto and Ghanem (2002).

The user can explicitly state the order used in the inversion of underlying Gaussian correlation kernel. The command is:

```
1 Hermite polynomial chaos Karhunen Loeve expansion to random field # <.> with ←
Hermite polynomial chaos dimension <.> order <.> ←
correlation_kernel_inverse_order = <.>;
```

Where:

- `dimension`: specifies the number of dimensions of Hermite polynomial chaos to capture the correlation structure of the random field
- `order`: specifies the order of Hermite polynomial chaos to capture the marginal distribution of the random field
- `correlation_kernel_inverse_order`: specifies the order used in the inversion of underlying Gaussian correlation kernel. For the exact Gaussian kernel inversion, set up `correlation_kernel_inverse_order` equal to order of Hermite polynomial chaos. `correlation_kernel_inverse_order` should not exceed order of Hermite polynomial chaos. If `correlation_kernel_inverse_order` is not stated, by default linear Gaussian kernel inversion, i.e., `correlation_kernel_inverse_order` equal to 1, is performed as the approximation of higher order inversion. See Sakamoto and Ghanem (2002) for more details.

For example:

```
1 Hermite polynomial chaos Karhunen Loeve expansion to random field # 1 with ←
Hermite polynomial chaos dimension 4 order 2 ←
correlation_kernel_inverse_order = 2;
```

Perform the Hermite polynomial chaos Karhunen Loève expansion for random field 1 using Hermite polynomial chaos of dimension 4 and order 2. The 2\textsuperscript{nd} order Gaussian correlation kernel inversion is adopted.
205.3.4.167  Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion Using HDF5 Input

Add a random field with marginal distribution and correlation information defined in a given HDF5 (.hdf5) file. Perform Hermite polynomial chaos Karhunen Loève expansion for the random field.

The command is:

```
Hermite polynomial chaos Karhunen Loève expansion to random field # <.> with ←
Hermite polynomial chaos dimension <.> order <.> hdf5_file = "file_name";
```

Where:

- **dimension**: specifies the number of dimensions of Hermite polynomial chaos to capture the correlation structure of the random field
- **order**: specifies the order of Hermite polynomial chaos to capture the marginal distribution of the random field
- **hdf5_file**: specifies the filename of the input HDF5 file that defines the marginal distribution and correlation information of the random field

The input HDF5 file should contain the following datasets:

- **Dataset Random Field** contains a single integer, which is the ID of the random field.
- **Dataset Marginal Mean** is a column vector specifying marginal mean of the random field corresponding to each random variable.
- **Dataset Marginal Variance** is a column vector specifying marginal variance of the random field for each random variable.
- **Dataset Marginal Distributions** is a column vector integers specifying the marginal distribution IDs of the random field for each random variable. Specifically, the ID is 1 for Gaussian distribution, 2 for Lognormal distribution, 3 for Gamma distribution and 4 for Weibull distribution.
- **Dataset Correlation** is a 2D array specifying correlation of the random field among random variables.
- **Dataset PC Order** contains a single integer, which specifies the order of Hermite polynomial chaos (PC) Karhunen Loève expansion.
• Dataset **PC Dimension** contains a single integer, which specifies the dimension of Hermite polynomial chaos (PC) Karhunen Loève expansion.

• Dataset **Index to Global Dimension** contains a column vector of integers, which specifies the index of corresponding global PC dimension for each local PC dimension used in the uncertainty expansion of the random field.

For example:

```
1 Hermite polynomial chaos Karhunen Loève expansion to random field # 1 with ←
   Hermite polynomial chaos dimension 4 order 4 hdf5_file = "PC_RF1.hdf5";
```

Perform the Hermite polynomial chaos Karhunen Loève expansion for random field 1 using Hermite polynomial chaos of dimension 4 and order 4 with input marginal distribution and correlation information defined in HDF5 file "PC_RF1.hdf5".
### 205.3.4.168 Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion with Inverse Order Using HDF5 Input

Add a random field with marginal distribution and correlation information defined in a given HDF5 (.hdf5) file. Perform Hermite polynomial chaos Karhunen Loève expansion for the random field.

The user can explicitly state the order used in the inversion of underlying Gaussian correlation kernel.

The command is:

```
Hermite polynomial chaos Karhunen Loève expansion to random field # <.> with ←
    Hermite polynomial chaos dimension <.> order <.> ←
    correlation_kernel_inverse_order = <.> hdf5_file = "file_name";
```

Where:

- **dimension**: specifies the number of dimensions of Hermite polynomial chaos to capture the correlation structure of the random field

- **order**: specifies the order of Hermite polynomial chaos to capture the marginal distribution of the random field

- **correlation_kernel_inverse_order**: specifies the order used in the inversion of underlying Gaussian correlation kernel. For the exact Gaussian kernel inversion, set up `correlation_kernel_inverse_order` equal to `order` of Hermite polynomial chaos. `correlation_kernel_inverse_order` should not exceed `order` of Hermite polynomial chaos. If `correlation_kernel_inverse_order` is not stated, by default linear Gaussian kernel inversion, i.e., `correlation_kernel_inverse_order` equal to 1, is performed as the approximation of higher order inversion. See Sakamoto and Ghanem (2002) for more details.

- **hdf5_file**: specifies the filename of the input HDF5 file that defines the marginal distribution and correlation information of the random field

The input HDF5 file should contain the following datasets:

- **Dataset Random Field** contains a single integer, which is the ID of the random field.

- **Dataset Marginal Mean** is a column vector specifying marginal mean of the random field corresponding to each random variable.

- **Dataset Marginal Variance** is a column vector specifying marginal variance of the random field for each random variable.
• Dataset **Marginal Distributions** is a column vector integers specifying the marginal distribution IDs of the random field for each random variable. Specifically, the ID is 1 for Gaussian distribution, 2 for Lognormal distribution, 3 for Gamma distribution and 4 for Weibull distribution.

• Dataset **Correlation** is a 2D array specifying correlation of the random field among random variables.

• Dataset **PC Order** contains a single integer, which specifies the order of Hermite polynomial chaos (PC) Karhunen Loève expansion.

• Dataset **PC Dimension** contains a single integer, which specifies the dimension of Hermite polynomial chaos (PC) Karhunen Loève expansion.

• Dataset **Index to Global Dimension** contains a column vector of integers, which specifies the index of corresponding global PC dimension for each local PC dimension used in the uncertainty expansion of the random field.

For example:

```
1 Hermite polynomial chaos Karhunen Loeve expansion to random field # 1 with ←
   Hermite polynomial chaos dimension 4 order 4 ←
   correlation_kernel_inverse_order = 3 hdf5_file = "PC_RF1.hdf5";
```

Perform the Hermite polynomial chaos Karhunen Loève expansion for random field 1 using Hermite polynomial chaos of dimension 4 and order 4. The $3^{rd}$ order Gaussian correlation kernel inversion is adopted. The input marginal distribution and correlation information are defined in HDF5 file "PC_RF1.hdf5".
205.3.4.169 Modeling, Random Field: Output Hermite Polynomial Chaos Karhunen Loève Expansion Result

A HDF5 (.hdf5) file contains all the information for Hermite polynomial chaos Karhunen Loève expansion for random field can be generated.

The command is:

```bash
1 generate Hermite polynomial chaos Karhunen Loeve expansion file from random field # <.> hdf5_file = "file_name";
```

where:

- `file_name` is a string that specifies the name of the output hdf5 file.

The generated hdf5 file contains the following datasets:

- **Dataset Random Field** contains a single integer, which is the ID of the random field.

- **Dataset Random Variables** contains a column vector of integers, which are the IDs of the random variables that constitute the random field.

- **Dataset Marginal Mean** is a column vector specifying marginal mean of the random field corresponding to each random variable.

- **Dataset Marginal Variance** is a column vector specifying marginal variance of the random field for each random variable.

- **Dataset Marginal Distributions** is a column vector integers specifying the marginal distribution IDs of the random field for each random variable. Specifically, the ID is 1 for Gaussian distribution, 2 for Lognormal distribution, 3 for Gamma distribution and 4 for Weibull distribution.

- **Dataset Correlation** is a 2D array specifying correlation of the random field among random variables.

- **Dataset PC Order** contains a single integer, which specifies the order of Hermite polynomial chaos (PC) Karhunen Loève expansion.

- **Dataset PC Dimension** contains a single integer, which specifies the dimension of Hermite polynomial chaos (PC) Karhunen Loève expansion.

- **Dataset Index to Global Dimension** contains a column vector of integers, which specifies the index of corresponding global PC dimension for each local PC dimension used in the uncertainty expansion of the random field.
• Dataset **PC** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis. \( PC_{ij} \) denotes the order of polynomial chaos dimension \( \xi_j \) that contributes to the \( i^{th} \) multidimensional Hermite PC basis.

• Dataset **PC Coefficients** is a 2D array. The \( j^{th} \) component of the \( i^{th} \) row is the polynomial chaos coefficient corresponding to the \( j^{th} \) PC base as described by **PC** for the \( i^{th} \) random variable specified in **Random Variables**.

• Dataset **PC Variance** is a column vector specifying the variances of Hermite PC basis.

For example:

```python
1 generate Hermite polynomial chaos Karhunen Loeve expansion file from random field # 1
field # 1 hdf5_file = "PC_RF1.hdf5";
```

Generate HDF5 file named “PC_RF1.hdf5” that contains all the information for Hermite polynomial chaos Karhunen Loève expansion of random field 1.
205.3.4.170 Modeling, Random Field: Adding Random Field from Hermite Polynomial Chaos Karhunen Loève Expansion HDF5 File

Add a random field with marginal distribution, correlation information and multi-dimensional Hermite polynomial chaos (PC) coefficients specified in a given HDF5 (.hdf5) file.

The command is:

```plaintext
1 add random field # <.> with Hermite polynomial chaos Karhunen Loeve expansion ↔
   hdf5_file = "file_name";
```

where:

- `file_name` is a string that specifies the name of the input hdf5 file.

The input hdf5 file should contain the following datasets:

- **Dataset Random Field** contains a single integer, that represents the ID of the random field.
- **Dataset Random Variables** contains a column vector of integers, that are the IDs of the random variables that constitute the random field.
- **Dataset Marginal Mean** is a column vector specifying marginal mean of the random field corresponding to each random variable.
- **Dataset Marginal Variance** is a column vector specifying marginal variance of the random field for each random variable.
- **Dataset Marginal Distributions** is a column vector of integers specifying the marginal distribution IDs of the random field for each random variable. Specifically, the ID is 1 for Gaussian distribution, 2 for Lognormal distribution, 3 for Gamma distribution and 4 for Weibull distribution.
- **Dataset Correlation** is a 2D array specifying correlation of the random field among random variables.
- **Dataset PC Order** contains a single integer, that specifies the order of Hermite polynomial chaos (PC) Karhunen Loève expansion.
- **Dataset PC Dimension** contains a single integer, that specifies the dimension of Hermite polynomial chaos (PC) Karhunen Loève expansion.
- **Dataset Index to Global Dimension** contains a column vector of integers, that specify the index of corresponding global PC dimension for each local PC dimension used in the uncertainty expansion of the random field.
• Dataset $\textbf{PC}$ is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis. $PC_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ multidimensional Hermite PC basis.

• Dataset $\textbf{PC Coefficients}$ is a 2D array. The $j^{th}$ component of the $i^{th}$ row is the polynomial chaos coefficient corresponding to the $j^{th}$ PC base as described by $\textbf{PC}$ for the $i^{th}$ random variable specified in $\textbf{Random Variables}$.

• Dataset $\textbf{PC Variance}$ is a column vector specifying the variances of Hermite PC basis.

For example:

```python
add random field # 1 with Hermite polynomial chaos Karhunen Loeve expansion
hdf5_file = "PC_RF1.hdf5";
```

Add random field 1 with the marginal distribution, correlation structure and Hermite polynomial chaos Karhunen Loève expansion information defined in HDF5 file named “PC_RF1.hdf5”.
205.3.4.171 Modeling, Random Field: Adding Random Field from Marginal Distribution and Correlation

Add a random field with specified marginal distribution and correlation information.

The command is:

```plaintext
1  add random field # <.> with <distribution_type> distribution
2  marginal_mean_file = "file_name"
3  marginal_standard_deviation_file = "file_name"
4  correlation_file = "file_name";
```

where:

- `distribution_type` is a string specifying the marginal distribution type of the random field, can be Gaussian, Lognormal, or Gamma.

- `marginal_mean_file` is a string specifying the name of a plain text file that contains a single column of marginal mean of the random field.

- `marginal_standard_deviation_file` is a string specifying the name of a plain text file that contains a single column of marginal standard deviation of the random field.

- `correlation_file` is a string specifying the name of a plain text file that contains a 2D array of the correlation structure of the random field.
205.3.4.172 Modeling, Random Field: Add Triple Product of Hermite Polynomial Chaos Basis

Compute and add triple product of Hermite polynomial chaos basis from three different random fields. The command is:

```
add triple product # <.> with Hermite polynomial chaos from random field (<.>, ←<.>, <.>);
```

For example:

```
add triple product # 1 with Hermite polynomial chaos from random field (1, 2, 3);
```

Compute and add triple product #1 with Hermite polynomial chaos basis from random field 1, 2 and 3;
205.3.4.173 Modeling, Random Field: Add Double Product of Hermite Polynomial Chaos Basis

Compute and add double product of Hermite polynomial chaos basis from two different random fields.

The command is:

```plaintext
1 add double product # <.> with Hermite polynomial chaos from random field (<.>, ← <.>);
```

For example:

```plaintext
1 add double product # 1 with Hermite polynomial chaos from random field (2, 3);
```

Compute and add double product #1 with Hermite polynomial chaos basis from random field 2 and 3;
205.3.4.174 Modeling, Random Field: Generate Triple Product of Hermite Polynomial Chaos Basis

The computation of triple product of Hermite polynomial chaos basis for different random fields is a key part for stochastic finite element analysis.

A HDF5 (.hdf5) file contains triple product of Hermite polynomial chaos basis for random fields can be generated.

The command is:

```
1 generate triple product of Hermite polynomial chaos from random field (<.,<.,<.>) hdf5_file = "file_name";
```

where:

• **file_name** is a string that specifies the name of the output HDF5 file.

The generated HDF5 file contains the following datasets:

• Dataset **PC1** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. \( PC1_{ij} \) denotes the order of polynomial chaos dimension \( \xi_j \) that contributes to the \( i^{th} \) basis of \( PC1 \).

• Dataset **PC1 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis \( PC1 \).

• Dataset **PC1 Variance** is a column vector specifying the variances of basis \( PC1 \).

• Dataset **PC2** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. \( PC2_{ij} \) denotes the order of polynomial chaos dimension \( \xi_j \) that contributes to the \( i^{th} \) basis of \( PC2 \).

• Dataset **PC2 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis \( PC2 \).

• Dataset **PC2 Variance** is a column vector specifying the variances of basis \( PC2 \).

• Dataset **PC3** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. \( PC3_{ij} \) denotes the order of polynomial chaos dimension \( \xi_j \) that contributes to the \( i^{th} \) basis of \( PC3 \).

• Dataset **PC3 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis \( PC3 \).
• Dataset **PC3 Variance** is a column vector specifying the variances of basis **PC3**.

• Dataset **Triple Product** is a column vector containing the non-zero triple products of polynomial chaos basis from **PC1**, **PC2** and **PC3**.

• Dataset **Triple Product PC1 Index** is a column vector containing the indexes of polynomial chaos basis from **PC1** that contributes to the non-zero triple products in dataset **Triple Product**.

• Dataset **Triple Product PC2 Index** is a column vector containing the indexes of polynomial chaos basis from **PC2** that contributes to the non-zero triple products in dataset **Triple Product**.

• Dataset **Triple Product PC3 Index** is a column vector containing the indexes of polynomial chaos basis from **PC3** that contributes to the non-zero triple products in dataset **Triple Product**.

For example:

```python
generate triple product of Hermite polynomial chaos from random field (1, 2, 3) ←
hdf5_file = "Triple_product_4(3)_4(3)_4(3).hdf5";
```

Compute the triple product of Hermite polynomial chaos basis of random field 1, 2 and 3 and write all the results into a HDF5 file named "Triple_product_4(3)_4(3)_4(3).hdf5";
205.3.4.175 Modeling, Random Field: Generate Double Product of Hermite Polynomial Chaos Basis

The computation of double product of Hermite polynomial chaos basis for different random fields is a key part for stochastic finite element analysis.

A HDF5 (.hdf5) file contains double product of Hermite polynomial chaos basis for random fields can be generated.

The command is:

```python
generate double product of Hermite polynomial chaos from random field (<.>, ←<.>) hdf5_file = "file_name";
```

where:

- `file_name` is a string that specifies the name of the output HDF5 file.

The generated HDF5 file contains the following datasets:

- **Dataset PC1** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. $PC1_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ basis of PC1.
- **Dataset PC1 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis PC1.
- **Dataset PC1 Variance** is a column vector specifying the variances of basis PC1.
- **Dataset PC2** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. $PC2_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ basis of PC2.
- **Dataset PC2 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis PC2.
- **Dataset PC2 Variance** is a column vector specifying the variances of basis PC2.
- **Dataset Double Product** is a column vector containing the non-zero double products of polynomial chaos basis from PC1 and PC2.
- **Dataset Double Product PC1 Index** is a column vector containing the indexes of polynomial chaos basis from PC1 that contributes to the non-zero double products in dataset Double Product.
• Dataset **Double Product PC2 Index** is a column vector containing the indexes of polynomial chaos basis from **PC2** that contributes to the non-zero double products in dataset **Double Product**.

For example:

```python
hdf5_file = "doubleproduct_153(2)_150(1).hdf5"
```

Compute the triple product of Hermite polynomial chaos basis of random field 1 and 2 and write all the results into a HDF5 file named "doubleproduct_153(2)_150(1).hdf5";
205.3.4.176  Modeling, Random Field: Add Triple Product of Hermite Polynomial Chaos Basis Using HDF5 Input

Add triple product of Hermite polynomial chaos basis using HDF5 input.

The command is:

```
add triple product # <.> from hdf5_file = "file_name";
```

where:

- `file_name` is a string that specifies the name of the input HDF5 file.

The input HDF5 file should contain the following datasets:

- Dataset **PC1** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. $PC1_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ basis of **PC1**.

- Dataset **PC1 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis **PC1**.

- Dataset **PC1 Variance** is a column vector specifying the variances of basis **PC1**.

- Dataset **PC2** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. $PC2_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ basis of **PC2**.

- Dataset **PC2 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis **PC2**.

- Dataset **PC2 Variance** is a column vector specifying the variances of basis **PC2**.

- Dataset **PC3** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. $PC3_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ basis of **PC3**.

- Dataset **PC3 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis **PC3**.

- Dataset **PC3 Variance** is a column vector specifying the variances of basis **PC3**.

- Dataset **Triple Product** is a column vector containing the non-zero triple products of polynomial chaos basis from **PC1**, **PC2** and **PC3**.
• Dataset **Triple Product PC1 Index** is a column vector containing the indexes of polynomial chaos basis from \( \text{PC1} \) that contributes to the non-zero triple products in dataset **Triple Product**.

• Dataset **Triple Product PC2 Index** is a column vector containing the indexes of polynomial chaos basis from \( \text{PC2} \) that contributes to the non-zero triple products in dataset **Triple Product**.

• Dataset **Triple Product PC3 Index** is a column vector containing the indexes of polynomial chaos basis from \( \text{PC3} \) that contributes to the non-zero triple products in dataset **Triple Product**.

For example:

```matlab
1 add triple product # 1 from hdf5_file = "tripleproduct_3(2)_153(2)_153(2).hdf5";
```

Add triple product #1 using HDF5 input file named "tripleproduct_3(2)_153(2)_153(2).hdf5".
205.3.4.177 Modeling, Random Field: Add Double Product of Hermite Polynomial Chaos Basis Using HDF5 Input

Add double product of Hermite polynomial chaos basis using HDF5 input.

The command is:

```
add double product # <.> from hdf5_file = "file_name";
```

where:

- `file_name` is a string that specifies the name of the input HDF5 file.

The input HDF5 file should contain the following datasets:

- **Dataset PC1** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. $PC1_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ basis of PC1.

- **Dataset PC1 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis PC1.

- **Dataset PC1 Variance** is a column vector specifying the variances of basis PC1.

- **Dataset PC2** is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) basis of the first random field. $PC2_{ij}$ denotes the order of polynomial chaos dimension $\xi_j$ that contributes to the $i^{th}$ basis of PC2.

- **Dataset PC2 Index to Global Dimension** contains a column vector of integers, which specifies the global dimension index of each local PC dimension in basis PC2.

- **Dataset PC2 Variance** is a column vector specifying the variances of basis PC2.

- **Dataset Double Product** is a column vector containing the non-zero double products of polynomial chaos basis from PC1 and PC2.

- **Dataset Double Product PC1 Index** is a column vector containing the indexes of polynomial chaos basis from PC1 that contributes to the non-zero double products in dataset Double Product.

- **Dataset Double Product PC2 Index** is a column vector containing the indexes of polynomial chaos basis from PC2 that contributes to the non-zero double products in dataset Double Product.
For example:

```python
add double product # 1 from hdf5_file = "doubleproduct_153(2)_150(1).hdf5";
```

Add double product #1 using HDF5 input file named "doubleproduct_153(2)_150(1).hdf5".
205.3.4.178 Modeling, Solid-Fluid Interaction: Adding Solid-Fluid Interface

For solid-fluid interaction analysis, solid fluid interface should be defined and added to the analysis domain.

The command is:

```plaintext
1 add solid fluid interface <string>
```

where:

- `<string>` specifies the name of the boundary of fluid domain that is solid fluid interface. It is noted that the boundary name should be consistent with the definition in the OpenFOAM input file at `constant/polyMesh/boundary`. More information about the organization and format of OpenFOAM input files can be found at OpenFOAM User Guide (OpenCFD Ltd, 2019).

For example:

```plaintext
1 add solid fluid interface "bottom_fluid_surface";
```

Adds fluid boundary named “bottom_fluid_surface” as one of the interface boundaries between solid domain and fluid domain.
205.3.4.179 Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface, ESSI Element Nodes

For solid-fluid interaction analysis, solid fluid interface contains information about the Real-ESSI solid interface and OpenFOAM fluid interface. The Real-ESSI solid interface defines Real-ESSI interface nodes and faces.

Real-ESSI interface nodes can be defined as the following:

```plaintext
define solid fluid interface ESSI nodes <string>;
```

where:

- `<string>` specifies the plain text file name that contains the information about Real-ESSI interface nodes.

The format of such text file containing Real-ESSI interface nodes information is:

- Comment line starts with “//”
- Each line defining a Real-ESSI interface node has four entries separated by space(s):
  - 1\textsuperscript{st} entry: Real-ESSI node ID;
  - 2\textsuperscript{nd} entry: x coordinate of Real-ESSI interface node;
  - 3\textsuperscript{rd} entry: y coordinate of Real-ESSI interface node;
  - 4\textsuperscript{th} entry: z coordinate of Real-ESSI interface node;

For example:

```plaintext
define solid fluid interface ESSI nodes "ESSI_nodes_info.fei";
```

Defines Real-ESSI nodes at solid fluid interface with file named “ESSI_nodes_info.fei”. An example file of “ESSI_nodes_info.fei” is provided below:

```plaintext
//====================================================
// Files contains information about ESSI nodes at solid fluid interface, have 4 ← columns
// 1st column: ESSI node ID
// 2nd column: coordinate x
// 3rd column: coordinate y
// 4th column: coordinate z
//====================================================
1 0.00 0.00 0.00
12 30.00 0.00 0.00
33 0.00 0.00 10.00
... 
```
205.3.4.180 Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface, ESSI Element Faces

For solid-fluid interaction analysis, solid fluid interface contains information about Real-ESSI solid interface and OpenFOAM fluid interface. The Real-ESSI solid interface defines Real-ESSI interface nodes and faces.

Real-ESSI interface faces are quads that can be defined as:

```
1 define solid fluid interface ESSI faces <string>; ...
```

where:

- `<string>` specifies the plain text file name that contains the information about Real-ESSI interface elements faces.

The format of such text file containing Real-ESSI interface element faces information is:

- Comment line starts with “//”

- Each line defining a Real-ESSI interface element face, i.e., a face of a brick element, in effect a quad consisting of four Real-ESSI interface nodes. Each line has six entries separated by spaces:
  - 1\(^{st}\) entry: Real-ESSI interface face ID;
  - 2\(^{nd}\) entry: Real-ESSI element ID that contains the Real-ESSI interface face;
  - 3\(^{rd}\) entry: Real-ESSI node ID for Real-ESSI interface face, i.e., a brick face, a quad, vertex 1;
  - 4\(^{th}\) entry: Real-ESSI node ID for Real-ESSI interface face, i.e., a brick face, a quad, vertex 2;
  - 5\(^{th}\) entry: Real-ESSI node ID for Real-ESSI interface face, i.e., a brick face, a quad, vertex 3;
  - 6\(^{th}\) entry: Real-ESSI node ID for Real-ESSI interface face, i.e., a brick face, a quad, vertex 4;

It is noted that the ordering of the four vertex Real-ESSI nodes is very important. The ordering should be taken such that the face normal vector points outwards to the fluid domain following the convention of right hand rule.

For example:
define solid fluid interface ESSI faces "ESSI_faces_info.fei";

Defines Real-ESSI faces at solid fluid interface with file named "ESSI_faces_info.fei". An example file of "ESSI_faces_info.fei" is provided below:

```plaintext
//================================================
// Files contains information about ESSI faces (quads) at solid fluid interface, have 6 columns
// 1st column: ESSI face ID
// 2nd column: ESSI Element ID that ESSI face belongs to
// 3rd column: ESSI node ID for quad vertex 1
// 4th column: ESSI node ID for quad vertex 2
// 5th column: ESSI node ID for quad vertex 3
// 6th column: ESSI node ID for quad vertex 4
//================================================
1 276 1 25 272 5
2 277 25 26 273 272
3 278 26 27 274 273
4 279 27 28 275 274
...```
205.3.4.181 Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface FOAM Nodes

For solid-fluid interaction analysis, solid fluid interface contains information about the Real-ESSI solid interface and OpenFOAM fluid interface. The OpenFOAM fluid interface needs to define OpenFOAM interface nodes and faces.

OpenFOAM interface nodes can be defined as the following:

```plaintext
define solid fluid interface FOAM nodes <string>;
```

where:

- `<string>` specifies the plain text file name that contains the information about OpenFOAM interface nodes.

The format of such text file containing OpenFOAM interface nodes information is:

- Comment line starts with `//`
- Each line defining a OpenFOAM interface node has four entries separated by space(s):
  - `1st` entry: OpenFOAM node ID;
  - `2nd` entry: x coordinate of OpenFOAM interface node;
  - `3rd` entry: y coordinate of OpenFOAM interface node;
  - `4th` entry: z coordinate of OpenFOAM interface node;

For example:

```plaintext
define solid fluid interface FOAM nodes "foam_nodes_info.fei";
```

Defines OpenFOAM nodes at solid fluid interface with file named "foam_nodes_info.fei". An example file of "foam_nodes_info.fei" can be:

```plaintext
//====================================================
// Files contains information about Foam nodes at solid fluid interface, have 4 <- columns
// 1st column: Foam node ID
// 2nd column: coordinate x
// 3rd column: coordinate y
// 4th column: coordinate z
//====================================================
1 0.00 0.00 0.00
12 30.00 0.00 0.00
33 0.00 0.00 10.00
...
```
205.3.4.182 Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface FOAM Faces

For solid-fluid interaction analysis, solid fluid interface contains information about the Real-ESSI solid interface and OpenFOAM fluid interface. The OpenFOAM fluid interface needs to define OpenFOAM interface nodes and faces.

OpenFOAM interface faces are quads that can be defined as the following:

```plaintext
1 define solid fluid interface FOAM faces <string>;
```

where:

- `<string>` specifies the plain text file name that contains the information about OpenFOAM interface faces.

The format of such text file containing OpenFOAM interface faces information is:

- Comment line starts with `//`
- Each line defining a OpenFOAM interface face, i.e., a quad consisting of four OpenFOAM interface nodes. Each line has five entries separated by space(s):
  - \(1^{st}\) entry: OpenFOAM face ID;
  - \(2^{nd}\) entry: OpenFOAM node ID for OpenFOAM interface face, i.e., a quad, vertex 1;
  - \(3^{rd}\) entry: OpenFOAM node ID for OpenFOAM interface face, i.e., a quad, vertex 2;
  - \(4^{th}\) entry: OpenFOAM node ID for OpenFOAM interface face, i.e., a quad, vertex 3;
  - \(5^{th}\) entry: OpenFOAM node ID for OpenFOAM interface face, i.e., a quad, vertex 4;

It is noted that the ordering of the four vertex OpenFOAM nodes is very important. The ordering should be consistent with the OpenFOAM faces definition in the OpenFOAM input file at `constant/polyMesh/faces`. Detailed information about the organization and format of OpenFOAM input files can be found at OpenFOAM User Guide (OpenCFD Ltd, 2019).

For example:

```plaintext
1 define solid fluid interface FOAM faces "foam_faces_info.fei";
```

Defines OpenFOAM faces at solid fluid interface with file named “foam_faces_info.fei”. An example file of “foam_faces_info.fei” can be:
// Files contains information about Foam faces (quads) at solid fluid interface, have 5 columns
// 1st column: Foam face ID
// 2nd column: Foam node ID for quad vertex 1
// 3th column: Foam node ID for quad vertex 2
// 4th column: Foam node ID for quad vertex 3
// 5th column: Foam node ID for quad vertex 4

50 8 0 4 404
51 9 8 404 405
52 10 9 405 406
53 11 10 406 407
54 12 11 407 408
55 13 12 408 409
...
205.3.5 Simulation
205.3.5.1 Simulation, Solvers: Sequential Solvers

```bash
define solver sequential <profilespd|umfpack>;
```

ProfileSPD and UMFPack are sequential solvers.

- ProfileSPD is used for symmetric matrices.
- UMFPack is used for asymmetric matrices and indefinite matrices.
205.3.5.2 Simulation, Solvers: Parallel Solvers

```plaintext
1  define solver parallel petsc <petsc_options> ;
```

**Direct Solvers**

Command Example for a direct solver:

```plaintext
1  define solver parallel petsc "-ksp_type preonly -pc_type lu" ;
2  define solver parallel petsc "-pc_type lu -pc_factor_mat_solver_package mumps" ;
3  define solver parallel petsc "-pc_type lu -pc_factor_mat_solver_package superlu" ;
```

As shown in the Command Example, "-ksp_type" represents the solver type, "-pc_type" represents the preconditioner types. By defining "preonly", petsc will use the direct solver, and its type is defined in the preconditioner types. In addition, "lu" represents LU factorization in the direct solver. The solver package "mumps" is designed for finite-element methods, and can interleave Gauss elimination process with the assembly process of global stiffness from the local element stiffness matrices. It is also noted that "mumps" can solve symmetric indefinite matrices.

The solver package "superlu" pivots the large-scale sparse matrices to numerous small-scale dense matrices for acceleration.

**Iterative Solvers**

Command Example for an iterative solver:

```plaintext
1  define solver parallel petsc "-ksp_type gmres -pc_type jacobi";
2  define solver parallel petsc "-ksp_type cg -pc_type ilu";
```

PETSc contains many iterative solvers and preconditioner for large-scale problems, and they are all available in with Real-ESSI.

Tables 205.1 and 205.2 on next pages present a full set of options for iterative solvers and preconditioners.
### Table 205.1: Available Parallel Iterative Solvers

<table>
<thead>
<tr>
<th>Solver Name</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;richardson&quot;</td>
<td>Richardson</td>
</tr>
<tr>
<td>&quot;chebyshev&quot;</td>
<td>Chebyshev</td>
</tr>
<tr>
<td>&quot;cg&quot;</td>
<td>Conjugate Gradient</td>
</tr>
<tr>
<td>&quot;bicg&quot;</td>
<td>BiConjugate Gradient</td>
</tr>
<tr>
<td>&quot;gmres&quot;</td>
<td>Generalized Minimal Residual</td>
</tr>
<tr>
<td>&quot;fgmres&quot;</td>
<td>Flexible Generalized Minimal Residual</td>
</tr>
<tr>
<td>&quot;dgmres&quot;</td>
<td>Deflated Generalized Minimal Residual</td>
</tr>
<tr>
<td>&quot;gcr&quot;</td>
<td>Generalized Conjugate Residual</td>
</tr>
<tr>
<td>&quot;bcgs&quot;</td>
<td>BiCGSTAB</td>
</tr>
<tr>
<td>&quot;cgs&quot;</td>
<td>Conjugate Gradient Squared</td>
</tr>
<tr>
<td>&quot;tfqmr&quot;</td>
<td>Transpose-Free Quasi-Minimal Residual (1)</td>
</tr>
<tr>
<td>&quot;tcqmr&quot;</td>
<td>Transpose-Free Quasi-Minimal Residual (2)</td>
</tr>
<tr>
<td>&quot;cr&quot;</td>
<td>Conjugate Residual</td>
</tr>
<tr>
<td>&quot;lsqr&quot;</td>
<td>Least Squares Method</td>
</tr>
</tbody>
</table>

### Table 205.2: Available Parallel Iterative Preconditioners

<table>
<thead>
<tr>
<th>Preconditioner Name</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>jacobi</td>
<td>Jacobi</td>
</tr>
<tr>
<td>bjacobi</td>
<td>Block Jacobi</td>
</tr>
<tr>
<td>sor</td>
<td>SOR (and SSOR)</td>
</tr>
<tr>
<td>eisenstat</td>
<td>SOR with Eisenstat trick</td>
</tr>
<tr>
<td>icc</td>
<td>Incomplete Cholesky</td>
</tr>
<tr>
<td>ilu</td>
<td>Incomplete LU</td>
</tr>
<tr>
<td>asm</td>
<td>Additive Schwarz</td>
</tr>
<tr>
<td>gasm</td>
<td>Generalized Additive Schwarz</td>
</tr>
<tr>
<td>gamg</td>
<td>Algebraic Multigrid</td>
</tr>
<tr>
<td>bddc</td>
<td>Balancing Domain Decomposition by Constraints</td>
</tr>
<tr>
<td>ksp</td>
<td>Linear solver</td>
</tr>
<tr>
<td>composite</td>
<td>Combination of preconditioners</td>
</tr>
</tbody>
</table>
205.3.5.3 Simulation: Static Solution Advancement

1. simulate <.> steps using static algorithm;
205.3.5.4 Simulation: Dynamic Solution Advancement with the Constant Time Step

```
simulate <.> steps using transient algorithm time_step = <T>;
```
205.3.5.5 Simulation: Dynamic Solution Advancement with Variable Time Step

```plaintext
simulate <.> steps using variable transient algorithm
  time_step = <T>
  minimum_time_step = <.>
  maximum_time_step = <.>
  number_of_iterations = <.>
```
205.3.5.6 Simulation: Generalized Eigenvalue Analysis

At any given point in an analysis a generalized eigenvalue analysis of the system can be performed, based on the current mass and tangent stiffness matrices. The command to do this is:

```
simulate using eigen algorithm number_of_modes = <.>;  
```

The first `number_of_modes` eigenvalues are displayed on screen after the analysis is performed. If more eigenvalues are requested than degrees-of-freedom the system has, the excess reported values are set to NaN (not a number).

Description of output for nodes of different dof types can be found in section 206.5.6.
205.3.5.7 Simulation: Displacement Control

```
define static integrator displacement_control using node # <.> dof DOFTYPE ←
    increment <L>;
```
205.3.5.8 Simulation: Load, Control, Factor Increment

1 \texttt{define load factor increment <.>;}
205.3.5.9 Simulation: Dynamic Integrator, Newmark Method

```plaintext
define dynamic integrator Newmark with gamma = <.> beta = <.>;
```
205.3.5.10  Simulation: Dynamic Integrator, Hilber Hughes Taylor, HHT, α Method

1  define dynamic integrator Hilber_Hughes_Taylor with alpha = <.;>;
205.3.5.11 Simulation: Absolute Convergence Criteria

```
define convergence test
< Absolute_Norm_Unbalanced_Force | Absolute_Norm_Displacement_Increment >
tolerance = <.>
maximum_iterations = <.>
```

This command sets the convergence criteria for global iterative solvers. If the system-of-equation to be solved is

\[ K_T \Delta U = \Delta R \]

where \( K_T \) is the current tangent stiffness operator/matrix, \( \Delta U \) is the displacement increment, and \( \Delta R \) is the residual. The convergence criteria is based on:

- The \( l^2 \) norm of the displacement increment: \( \| \Delta U \|_2 < TOL \).
- The \( l^2 \) norm of the unbalanced force: \( \| \Delta R \|_2 < TOL \).

The convergence test should be defined before the algorithms.
205.3.5.12 Simulation: Average Convergence Criteria

```plaintext
1  define convergence test ←
   <Average_Norm_Unbalanced_Force|Average_Norm_Displacement_Increment>
2  tolerance = <.>
3  maximum_iterations = <.> ;
```

This command sets the convergence criteria for global iterative solvers. If the system-of-equation to be solved is

\[ K_T \Delta U = \Delta R \]

Where \( K_T \) is the current tangent stiffness operator (dynamic tangent for dynamic analysis), \( \Delta U \) is the displacement increment, and \( \Delta R \) is the residual. The convergence criteria can be based off

- The average \( l^2 \) norm of the displacement increment: \( \| \Delta U \|_2 / \sqrt{N} < TOL \).
- The average \( l^2 \) norm of the unbalanced force: \( \| \Delta R \|_2 / \sqrt{N} < TOL \).

where \( N \) is the number of DOFs in the system-of-equations.

The convergence test should be defined before the algorithms.
205.3.5.13 Simulation: Relative Convergence Criteria

```plaintext
1 define convergence test ←
    <Relative_Norm_Unbalanced_Force|Relative_Norm_Displacement_Increment>
2 tolerance = <.>
3 minimum_absolute_tolerance = <.>
4 maximum_iterations = <.> ;
```

This command sets the convergence criteria for global iterative solvers. If the system-of-equation to be solved is

\[ K_T \Delta U = \Delta R \]

Where \( K_T \) is the current tangent stiffness operator (dynamic tangent for dynamic analysis), \( \Delta U \) is the displacement increment, and \( \Delta R \) is the residual. The convergence criteria can be based on

- The relative \( l^2 \) norm of the displacement increment: \( \|\Delta U\|_2/\|U_0\|_2 < TOL \) or \( \|\Delta U\|_2 < MIN\_ABS\_TOL \).

- The relative \( l^2 \) norm of the unbalanced force: \( \|\Delta R\|_2/\|R_0\|_2 < TOL \) or \( \|\Delta R\|_2 < MIN\_ABS\_TOL \).

Where,

- \( R_0 \) is the external force in the beginning
- \( U_0 \) is the solution after the first iteration.

Since \( U_0 \) is zero before the first iteration, the relative norm of the displacement increment in the first iteration would be equal to 1.

The convergence test should be defined before the algorithms.
205.3.5.14 Simulation: Solution Algorithms

```plaintext
define algorithm < With_no_convergence_check | linear_elastic | Newton | ←
    Modified_Newton | Newton_With_LineSearch >;

define algorithm < Newton_With_Subincrement >
    using minimum_time_step = <.> ;
```

If the current specified load factor $\Delta \lambda$ (for static) or time step $\Delta t$ (for dynamic) fails to achieve the convergence in the specified maximum number of iterations, the algorithm `Newton_With_Subincrement` will subdivide the current step into two sub steps of load increment $\Delta \lambda^{\text{new}} = \Delta \lambda / 2$ (for static) or time step $\Delta t^{\text{new}} = \Delta t / 2$ (for dynamic).

- `minimum_time_step` specifies the allowed minimum load factor $\Delta \lambda$ (for static) or time step $\Delta t$ (for dynamic), that the algorithm should sub-divide to achieve convergence. If the subdivided step size becomes less than the `minimum_time_step`, the algorithm returns failure to convergence.

**Note:** If any Newton algorithm is used, the convergence test should be defined before the algorithms.
205.3.5.15 Simulation: Constitutive Integration Algorithm

Starting with version 03–NOV-2015, NDMaterial class of materials require explicit specification of the constitutive integration algorithm. This is done with the command:

```csharp
1 define NDMaterial constitutive integration algorithm Forward_Euler;
```

```csharp
1 define NDMaterial constitutive integration algorithm Forward_Euler_Subincrement
2 number_of_subincrements = <.> ;
```

```csharp
1 define NDMaterial constitutive integration algorithm Backward_Euler
2 yield_function_relative_tolerance = <.>
3 stress_relative_tolerance = <.>
4 maximum_iterations = <.> ;
```

```csharp
1 define NDMaterial constitutive integration algorithm Backward_Euler_Subincrement
2 yield_function_relative_tolerance = <.>
3 stress_relative_tolerance = <.>
4 maximum_iterations = <.>
5 allowed_subincrement_strain = <.> ;
```

The command specifies the method, tolerances and maximum number of iterations used to do material point integrations. The parameters are:

- `number_of_subincrements` Specify the number of subincrements in forward Euler subincrement algorithm.
- `yield_function_relative_tolerance` Specify the relative tolerance of the yield surface value in the family of backward Euler algorithm.
- `stress_relative_tolerance` Specify the relative stress tolerance in the family of backward Euler algorithm. The stress increment is within this tolerance for each step unless the integration fails. Frobenius norm is used to calculate the stress norm.
- `maximum_iterations` Specify the maximum number of iterations in backward Euler algorithm.
- `allowed_subincrement_strain` defines the maximum value of allowed strain increment in backward Euler subincrement method. If one of strain component increments is greater than the user-defined allowed strain increment, strain increment will be divided into subincrements based on the allowed subincrement. For example, if the strain increment is 0.05, and the allowed_subincrement_strain is 0.01. The number of subincrements will be 0.05/0.01 = 5. A small allowed subincrement leads to more accurate results, however, it takes more time. For the simple nonlinear materials, like
von Mises linear hardening, the allowed subincrement can be as big as 5 percent. For the complicated nonlinear materials, like hyperbolic Drucker-Prager Armstrong-Frederick hardening material, the allowed subincrement should be much smaller in the range of 1E-4.
205.3.5.16 Simulation: Status Check

All simulate commands set the variable SIMULATE_EXIT_FLAG automatically upon exit. This flag can be used to check whether the simulation concluded normally (SIMULATE_EXIT_FLAG = 0), failed (SIMULATE_EXIT_FLAG < 0), or finished with warnings (SIMULATE_EXIT_FLAG > 0).

For example, the following simulations will fail.

```plaintext
atmospheric_pressure = 101325*Pa;
pstart = 3000*kPa;

//SANiSand 2004 calibration for Toyoura Sand.
add material # 1 type sanisand2004
  mass_density = 2100.0 * kg / m^3
  e0 = 0.735
  sanisand2004_G0 = 125. poisson_ratio = 0.05
  sanisand2004_Pat = atmospheric_pressure
  sanisand2004_p_cut = 0.1*atmospheric_pressure
  sanisand2004_Mc = 1.25 sanisand2004_c = 0.712
  sanisand2004_lambda_c = 0.019 sanisand2004_xi = 0.7
  sanisand2004_ec_ref = 0.934 sanisand2004_m = 0.01
  sanisand2004_h0 = 7.05 sanisand2004_ch = 0.968
  sanisand2004_nb = 1.1 sanisand2004_A0 = 0.704
  sanisand2004_nd = 3.5 sanisand2004_z_max = 4.
  sanisand2004_cz = 600.
  initial_confining_stress = 1*Pa

simulate constitutive testing DIRECT_STRAIN
  use material # 1
  scale_factor = 1.
  series_file = "increments.txt"
  sigma0 = ( -pstart*kPa , -pstart*kPa , -pstart*kPa , 0*Pa , 0*Pa , 0*Pa )
  verbose_output = 1;

if(SIMULATE_EXIT_FLAG == 0)
  {
    print "All Good!";
  }
else
  {
    print "Something went wrong. Error code = ";
    print SIMULATE_EXIT_FLAG;
  }
bye;
```

The above simulation fails because the integration method for the constitutive model is not set (see 205.3.5.15). Therefore, the second branch of the 'if' statement will execute.
205.3.5.17 Simulation: Save State

Save ESSI system state to a file, to prepare for a restart.

```
1 save model;
```

This command will save the state of a model, an ESSI system, in file. Filename for the save file will be created from a model name, loading stage name, and current loading time.

For example, for a model that contains the following commands in the input file:

```
1 model name "ESSI_model";
2 ...
3 new loading stage "Loading_stage_2";
4 ...
5 simulate 100 steps using transient algorithm time_step = 0.005*s;
6 ...
7 save model;
```

will be saved in file with the following name:

```
1 ESSI_model_Loading_stage_2_at_time_0.5second_RESTART.essi
```

since there were 100 steps with $\Delta t = 0.005\text{s}$, and that advances the solution to $t = 0.5\text{s}$.
205.3.5.18 Simulation: Restart Simulation

Restart simulation after stage or a step, from a saved ESSI system model file.

```plaintext
1 restart model using file "filename";
```

Here, ESSI system model is saved in a file `filename`, see command for saving model on page 1078.

For example, to restart simulation from a saved file described above, restart will be initiated in a new input file by using the following command:

```plaintext
1 restart model using file "ESSI_model_Loading_stage_2_at_time_0.5second_RESTART.essi";
```

All the results from previous loading stages will be saved in the restart file. From here on, analyst can start new loading stages, etc.
205.3.5.19 Simulation: Return Value for simulate Command

Simulate command, simulate, returns status of simulation progress. For each successful step, simulate returns value 0 while for a failed step it returns −1. This is useful as analyst can control solution process, and change algorithm if predefined algorithm fails to converge.

For example the example if listing 205.8, simulation part of a larger examples, will perform a change of stepping algorithm from the load control to displacement control upon failure of load control to converge.

```
step=0;
Nsteps = 100;
define load factor increment 0.01; // Start with load-control

simulation_status=simulate 1 steps using static algorithm;

while (step<(Nsteps-1))
{
    if(simulation_status>=0) // Converged, continue using load-control
    {
        simulation_status=simulate 1 steps using static algorithm;
    }
    else // Not converged, so change to displacement-control
    {
        define static integrator displacement_control using node # 1 dof ux ←
        increment 1E-3*m;
        simulate 1 steps using static algorithm;
    }
    step=step+1;
}
```

Figure 205.8: Interactive simulation control using feedback (return value) from the simulate command.

It should be noted that the idea for interactive control of simulation process comes from FEAP (Zienkiewicz and Taylor, 1991b) and later from OpenSEES (Mazzoni et al., 2002) where it was implemented with early extension of OpenSees command language with using Tcl in early 2000s.

An example of the above feedback mechanism is provided below
model name "vm";
add material # 1 type vonMises
  mass_density = 0.0*kg/m^3
  elastic_modulus = 2E7*N/m^2
  poisson_ratio = 0.0
  von_mises_radius = 1E5*Pa
  kinematic_hardening_rate = 0*Pa
  isotropic_hardening_rate = 0*Pa;
// define the node:
add node # 1 at (0*m,0*m,1*m) with 3 dofs;
add node # 2 at (1*m,0*m,1*m) with 3 dofs;
add node # 3 at (1*m,1*m,1*m) with 3 dofs;
add node # 4 at (0*m,1*m,1*m) with 3 dofs;
add node # 5 at (0*m,0*m,0*m) with 3 dofs;
add node # 6 at (1*m,0*m,0*m) with 3 dofs;
add node # 7 at (1*m,1*m,0*m) with 3 dofs;
add node # 8 at (0*m,1*m,0*m) with 3 dofs;
// Define the element.
add element # 1 type 8NodeBrick using 2 Gauss points each direction with nodes ↔
  (1, 2, 3, 4, 5, 6, 7, 8) use material # 1;

new loading stage "shearing";
//fix the bottom totally
fix node # 5 dofs all;
fix node # 6 dofs all;
fix node # 7 dofs all;
fix node # 8 dofs all;
// Fix the other 2 directions on the top.
fix node # 1 dofs uy uz ;
fix node # 2 dofs uy uz ;
fix node # 3 dofs uy uz ;
fix node # 4 dofs uy uz ;
add load # 101 to node # 1 type linear Fx = 40 * kN;
add load # 102 to node # 2 type linear Fx = 40 * kN;
add load # 103 to node # 3 type linear Fx = 40 * kN;
add load # 104 to node # 4 type linear Fx = 40 * kN;
define solver UMFPack;
//define algorithm With_no_convergence_check ↔
;Norm_Displacement_Increment;Norm_Unbalance
define convergence test Absolute_Norm_Displacement_Increment
tolerance = 1E-3
maximum_iterations = 5
;
define algorithm Newton;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-7
  stress_relative_tolerance = 1E-7
  maximum_iterations = 100;

// ***************************************************************************
step=0;
Nsteps = 10;
define load factor increment 1/Nsteps; // Start with load-control

// Simulate with status check:
//***************************************************************************
mystatus=simulate 1 steps using static algorithm;
while (step<(Nsteps-1)){
  step=step+1;
  if(mystatus>=0){ // Converged, so continue using load-control
    mystatus=simulate 1 steps using static algorithm;
  } // Not converged, so change to displacement-control
  define static integrator displacement_control using node # 1 dof ux ← increment 1E-3*m ;
      simulate 1 steps using static algorithm;
}
}
bye;

Resulting terminal output, showing a switch between two solution control mechanisms is provided below:

The Finite Element Interpreter
MS ESSI
Earthquake Soil Structure Interaction Simulator
Sequential processing mode.

Version Branch : yuan
Compile Date : Apr 15 2017 at 20:28:11
Compile User : yuan
Compile Sysinfo: cml01 4.4.0-72-generic x86_64 GNU/Linux

Static startup tips:
* Remember: Every command ends with a semicolon '\;'.
* Type 'quit;' or 'exit;' to finish.
* Run 'essi -h' to see available command line options.

Including: "main.fei"

Model name is being set to "vm"
Starting new stage: shearing

changing previous stage_name from to shearing

Setting set_constitutive_integration_method = 2

Starting sequential static multi-step analysis

Creating analysis ←
  model.................................................................Pass!
  Checking constraint ←
    handler...............................................................Pass!
  Checking ←
    numberer.............................................................Pass!
  Checking analysis ←
    algorithm..........................................................Pass!
  Checking system of equation ←
    handler.............................................................Pass!
  Checking static integration ←
    handler.............................................................Pass!

Writing Initial Conditions and (0) - Outputting mesh.

Static Analysis: [ 1/1 ]

[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement ←
  Increment::(tol: 0.001)
    Absolute Norm deltaF: 1.6396e-12
    Absolute Norm deltaU: 0.0032
    Average Norm deltaF: 8.1981e-13
    Average Norm deltaU: 0.0016
    Relative Norm deltaF: 2.0495e-16
    Relative Norm deltaU: 0.0032

[iteration 2 /5 ] Convergence Test: Absolute Norm Displacement ←
  Increment::(tol: 0.001)
    Absolute Norm deltaF: 4.5475e-13
    Absolute Norm deltaU: 5.2683e-19
    Average Norm deltaF: 2.2737e-13
    Average Norm deltaU: 2.6341e-19
    Relative Norm deltaF: 5.6843e-17
    Relative Norm deltaU: 1.6463e-16

> Analysis End ←

Starting sequential static multi-step analysis

Creating analysis ←
  model.................................................................Pass!
  Checking constraint ←
handler............................................................Pass!
Checking numberer............................................................Pass!
Checking algorithm............................................................Pass!
Checking system of equation handler............................................................Pass!
Checking static integration handler............................................................Pass!

Static Analysis: [1 /1 ]
[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement ←
Increment::(tol: 0.001)
Absolute Norm deltaF: 2.7285e-12
Absolute Norm deltaU: 0.0032
Average Norm deltaF: 1.3642e-12
Average Norm deltaU: 0.0016
Relative Norm deltaF: 3.4106e-16
Relative Norm deltaU: 0.0032

[iteration 2 /5 ] Convergence Test: Absolute Norm Displacement ←
Increment::(tol: 0.001)
Absolute Norm deltaF: 3.5225e-12
Absolute Norm deltaU: 7.8429e-19
Average Norm deltaF: 1.7612e-12
Average Norm deltaU: 3.9214e-19
Relative Norm deltaF: 4.4031e-16
Relative Norm deltaU: 2.4509e-16

> Analysis End ←

Starting sequential static multi-step analysis

Creating analysis ←
model............................................................Pass!
Checking constraint ←
handler............................................................Pass!
Checking numberer............................................................Pass!
Checking algorithm............................................................Pass!
Checking system of equation handler............................................................Pass!
Checking static integration handler............................................................Pass!

Static Analysis: [1 /1 ]
[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement ←
Increment::(tol: 0.001)
Absolute Norm deltaF: 1.819e-12
Absolute Norm deltaU: 0.0032
Average Norm deltaF: 9.0949e-13
Average Norm deltaU: 0.0016
Relative Norm deltaF: 2.2737e-16
Relative Norm deltaU: 0.0032

[iteration 2 /5 ] Convergence Test: Absolute Norm Displacement ←
Increment::(tol: 0.001)
Absolute Norm deltaF: 2.5724e-12
Absolute Norm deltaU: 5.0807e-19
Average Norm deltaF: 1.2862e-12
Average Norm deltaU: 2.5403e-19
Relative Norm deltaF: 3.2155e-16
Relative Norm deltaU: 1.5877e-16

> Analysis End ←

Starting sequential static multi-step analysis
Creating analysis ←
  model.................................................................Pass!
Checking constraint ←
  handler...............................................................Pass!
Checking ←
  numberer..............................................................Pass!
Checking analysis ←
  algorithm............................................................Pass!
Checking system of equation ←
  handler...............................................................Pass!
Checking static integration ←
  handler...............................................................Pass!

Static Analysis: [1 /1 ]
[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement ←
Increment::(tol: 0.001)
  Absolute Norm deltaF: 3132.5
  Absolute Norm deltaU: 0.0032
  Average Norm deltaF: 1566.2
  Average Norm deltaU: 0.0016
  Relative Norm deltaF: 0.39156
  Relative Norm deltaU: 0.0032

[iteration 2 /5 ] Convergence Test: Absolute Norm Displacement ←
Increment::(tol: 0.001)
  Absolute Norm deltaF: 3132.5
  Absolute Norm deltaU: 0.001253
  Average Norm deltaF: 1566.2
  Average Norm deltaU: 0.0006265
  Relative Norm deltaF: 0.39156
  Relative Norm deltaU: 0.39156

[iteration 3 /5 ] Convergence Test: Absolute Norm Displacement ←
Increment::(tol: 0.001)
  Absolute Norm deltaF: 3132.5
Absolute Norm deltaU: 0.001253
Average Norm deltaF: 1566.2
Average Norm deltaU: 0.0006265
Relative Norm deltaF: 0.39156
Relative Norm deltaU: 0.39156

[iteration 4 /5 ] Convergence Test: Absolute Norm Displacement $\leftarrow$
Increment::(tol: 0.001)
Absolute Norm deltaF: 3132.5
Absolute Norm deltaU: 0.001253
Average Norm deltaF: 1566.2
Average Norm deltaU: 0.0006265
Relative Norm deltaF: 0.39156
Relative Norm deltaU: 0.39156

[iteration 5 /5 ] Convergence Test: Absolute Norm Displacement $\leftarrow$
Increment::(tol: 0.001) !!!FAILED TO CONVERGE!!! [EXITING..]
Absolute Norm deltaF: 3132.5
Absolute Norm deltaU: 0.001253
Average Norm deltaF: 1566.2
Average Norm deltaU: 0.0006265
Relative Norm deltaF: 0.39156
Relative Norm deltaU: 0.39156

NewtonRaphson::solveCurrentStep() -the ConvergenceTest object failed in test()

Static Analysis: [1 /1 ] The Algorithm failed at load factor 0.4
> Analysis End $\leftarrow$

Starting sequential static multistep analysis

Creating analysis $\leftarrow$
  model.................................................................Pass!
Checking constraint $\leftarrow$
  handler.................................................................Pass!
Checking $\leftarrow$
  numberer.................................................................Pass!
Checking analysis $\leftarrow$
  algorithm.................................................................Pass!
Checking system of equation $\leftarrow$
  handler.................................................................Pass!
Checking static integration $\leftarrow$
  handler.................................................................Pass!

Static Analysis: [1 /1 ]
[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement

Increment::(tol: 0.001)

Absolute Norm deltaF: 9310.6
Absolute Norm deltaU: 2.6478e-19
Average Norm deltaF: 4655.3
Average Norm deltaU: 1.3239e-19
Relative Norm deltaF: 0.42857
Relative Norm deltaU: 2.6478e-19

> Analysis End

Starting sequential static multistep analysis

Creating analysis
  model.................................................................Pass!
  Checking constraint
    handler............................................................Pass!
  Checking
    numberer..............................................................Pass!
  Checking analysis
    algorithm...............................................................Pass!
  Checking system of equation
    handler............................................................Pass!
  Checking static integration
    handler............................................................Pass!

Static Analysis: [1 /1 ]

[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement

Increment::(tol: 0.001)

Absolute Norm deltaF: 1437.5
Absolute Norm deltaU: 1.0012e-18
Average Norm deltaF: 718.76
Average Norm deltaU: 5.006e-19
Relative Norm deltaF: 0.10425
Relative Norm deltaU: 1.0012e-18

> Analysis End

Starting sequential static multistep analysis

Creating analysis
  model.................................................................Pass!
  Checking constraint
    handler............................................................Pass!
  Checking
    numberer..............................................................Pass!
  Checking analysis
    algorithm...............................................................Pass!
  Checking system of equation

Checking static integration
   handler............................................................Pass!

Static Analysis: [1 /1 ]
   [iteration 1 /5 ] Convergence Test: Absolute Norm Displacement
      Increment::(tol: 0.001)
         Absolute Norm deltaF: 112.08
         Absolute Norm deltaU: 5.1789e-19
         Average Norm deltaF: 56.038
         Average Norm deltaU: 2.5895e-19
         Relative Norm deltaF: 0.017652
         Relative Norm deltaU: 5.1789e-19

> Analysis End

Starting sequential static multistep analysis

Creating analysis
   model........................................................................Pass!
   Checking constraint
      handler....................................................................Pass!
   Checking numberer......................................................................Pass!
   Checking analysis
      algorithm....................................................................Pass!
   Checking system of equation
      handler............................................................Pass!
   Checking static integration
      handler............................................................Pass!

Static Analysis: [1 /1 ]
   [iteration 1 /5 ] Convergence Test: Absolute Norm Displacement
      Increment::(tol: 0.001)
         Absolute Norm deltaF: 7.142
         Absolute Norm deltaU: 8.6792e-19
         Average Norm deltaF: 3.571
         Average Norm deltaU: 4.3396e-19
         Relative Norm deltaF: 0.001399
         Relative Norm deltaU: 8.6792e-19

> Analysis End

Starting sequential static multistep analysis

Creating analysis
   model........................................................................Pass!
   Checking constraint
      handler....................................................................Pass!
   Checking numberer......................................................................Pass!
   Checking analysis
      algorithm....................................................................Pass!
Checking
   numberer.................................................................Pass!
Checking analysis
   algorithm.................................................................Pass!
Checking system of equation
   handler.................................................................Pass!
Checking static integration
   handler.................................................................Pass!

Static Analysis: [1 /1 ]
[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement
   Increment::(tol: 0.001)
   Absolute Norm deltaF: 0.44693
   Absolute Norm deltaU: 1.0155e-18
   Average Norm deltaF: 0.22347
   Average Norm deltaU: 5.0774e-19
   Relative Norm deltaF: 8.9267e-05
   Relative Norm deltaU: 1.0155e-18

> Analysis End

Starting sequential static multistep analysis
====================================================================================================
Creating analysis
   model.................................................................Pass!
Checking constraint
   handler.................................................................Pass!
Checking
   numberer.................................................................Pass!
Checking analysis
   algorithm.................................................................Pass!
Checking system of equation
   handler.................................................................Pass!
Checking static integration
   handler.................................................................Pass!

Static Analysis: [1 /1 ]
[iteration 1 /5 ] Convergence Test: Absolute Norm Displacement
   Increment::(tol: 0.001)
   Absolute Norm deltaF: 0.027935
   Absolute Norm deltaU: 2.1658e-19
   Average Norm deltaF: 0.013968
   Average Norm deltaU: 1.0829e-19
   Relative Norm deltaF: 5.5866e-06
   Relative Norm deltaU: 2.1658e-19

> Analysis End

How polite! Bye, have a nice day!
205.3.20 Simulation: New Elastic Loading Case

For design applications, linear elastic analysis cases are performed and later combined, using factors of safety (see section 205.3.21 on page 1091) to obtain sectional forces for design.

The command for elastic analysis is:

1 new elastic loading case <string> ;

One example is

1 new elastic loading case "case1" ;

In a new elastic loading case, all previous loads, load patterns are removed.
To guarantee a fresh start, all commit-displacement at nodes are reset to 0, and all commit-stress/strain at Gauss points are reset to 0.
The following components are kept unchanged in a new elastic loading case:

- material properties.
- mesh connectivity
- boundary conditions.
- acceleration fields.
- damping.

If users want to modify the mesh, a new model is suggested instead of a new elastic loading case.
205.3.5.21 Simulation: Combine Elastic Load Cases

For design applications, elastic load cases, that have been analyzed beforehand, can be superimposed, combined using factors of safety, to obtain internal forces that used for design.

The command for this is

```plaintext
combine elastic load cases
hdf5_filenames_list = <string>
load_factors_list = <string>
output_filename = <string>
;
```

One example is

```plaintext
combine elastic load cases
hdf5_filenames_list = "test_case1.h5.feioutput test_case2.h5.feioutput"
load_factors_list = "1.2 1.5"
output_filename = "combine.h5.feioutput"
;
```

- `hdf5_filenames_list` specifies the list of HDF5 output filenames. The list should be separated by either space or comma.

- `load_factors_list` specifies the list of scale factors for each loading case. The list should be separated by either space or comma.

- `output_filename` specifies one output filename of the combined loading cases.

The number of specified files in `hdf5_filenames_list` should be equal to the number of scale factors (factors of safety) in `load_factors_list`.

205.3.5.22 Simulation, Dynamic Solution Advancement for Solid-Fluid Interaction

Dynamic analysis of solid-fluid interaction can be performed using:

```plaintext
simulate <.> steps using solid fluid interaction transient algorithm time_step ← = <T>;
```

where:

- `<.>` is an integer specifying total number of time steps in transient solid fluid interaction analysis.
- `time_step = <T>` defines the time step for solid fluid interaction transient analysis.

For example:

```plaintext
simulate 300 steps using solid fluid interaction transient algorithm time_step ← = 0.01*s;
```

Performs transient solid fluid interaction analysis for 300 steps with time step 0.01 s.
**205.3.5.23 Simulation, Dynamic Solution Advancement for Stochastic Finite Element Method**

Dynamic analysis of stochastic finite element modeling can be performed using:

1. `simulate <.> steps using stochastic transient algorithm time_step = <T>;`

where:

- `<.>` is an integer specifying total number of time steps in transient stochastic finite element analysis.
- `time_step = <T>` defines the time step for stochastic finite element transient analysis.

Please note that the stochastic transient algorithm is different from the general transient algorithm in section 205.3.5.4 in the formulation of unbalanced load for each time step. The stochastic transient algorithm uses directly the incremental external loads to compute the incremental displacements, while the general transient algorithm accounts for the correction from resisting forces in the formulation of unbalanced forces.

For deterministic and probabilistic, linear elastic problems, both transient algorithms would produce the same response. Stochastic transient algorithm is more efficient because there is no need to compute resisting forces.

For deterministic, nonlinear inelastic problems, the general transient algorithm is more accurate due to the corrections from resisting forces. The accuracy of stochastic transient algorithm can be improved using smaller loading increments. For probabilistic, nonlinear inelastic problems, the accuracy of the general transient algorithm can only be guaranteed if the number of polynomial chaos terms used in probabilistic constitutive modeling is equal or close enough to the number of polynomial chaos terms in global level. Otherwise, it is recommended to use the stochastic transient algorithm for dynamic analysis of stochastic finite element modeling.

For example:

1. `simulate 300 steps using stochastic transient algorithm time_step = 0.01*s;`

Performs transient stochastic finite element analysis for 300 steps with time step 0.01s.
205.3.5.24 Simulation, Sobol Sensitivity Analysis

Sobol sensitivity analysis can be performed using:

1. Sobol sensitivity analysis of node # <.> dof DOFTYPE peak response from random field # <.>
2. pc_coefficient_hdf5 = "pc_coefficient_hdf5_file_name"
3. output_hdf5 = "output_hdf5_file_name"

where:

- node # specify the node tag.
- DOFTYPE specify the dof to perform sensitivity analysis. It can be either \( u_x \), \( u_y \) or \( u_z \).
- random field # specify the random field polynomial chaos bases of the stochastic nodal response.
- pc_coefficient_hdf5 specify the name of a hdf5 file that contains simulation results of polynomial chaos coefficients.
- output_hdf5 specify the name of the output hdf5 file for sensitivity analysis.

The output hdf5 format for sensitivity analysis is given as below:

![sensitivity_result.hdf5]

- "Generalized_Accelerations_Sensitivity"
- "Generalized_Displacements_Sensitivity"
- "Index_to_Global_Dimension"
- "PC"
- "PC_Variance"
- "Sensitivity_Dimension_Groups"

Figure 205.9: Overall data structure of output hdf5 file for sensitivity analysis.

Figure 205.9 shows the overall data structure organization of the output hdf5 file of sensitivity analysis.

- **Generalized_Accelerations_Sensitivity** data group:

  Contains Sobol sensitivity analysis results for stochastic nodal acceleration response. It contains the following datasets and data groups as shown in Figure 205.10.
Figure 205.10: Datasets and data groups in Generalized_Accelerations_Sensitivity data group.

- **Component_Sobol_Indexes** dataset:
  Is a column vector containing the computed Sobol Indexes for each component of polynomial chaos (PC) bases.

- **Component_Sobol_Indexes_Sort** dataset:
  Is a column vector containing the computed Sobol Indexes for each component of polynomial chaos (PC) bases, in descending order.

- **Component_Variance_PC_Bases** dataset:
  Is a column vector containing the computed variance for each component of polynomial chaos (PC) bases.

- **PC_Coefficients** dataset:
  Is a column vector containing the polynomial chaos coefficients corresponding to PC bases specified in PC dataset.

- **PC_Sort** dataset:
  Is a 2D array that describes the sorted PC bases corresponding to Component_Sobol_Indexes_Sort dataset. \( PC\_Sort_{ij} \) denotes the order of polynomial chaos dimension \( \xi_j \) that contributes to the \( i^{th} \) sorted, PC basis.

- **Total_Variance** dataset:
  Is a scalar, variance of nodal stochastic response.

In addition to the above datasets, there will be sub data group(s) containing sensitivity analysis results corresponding to each of the defined sensitivity dimension groups. For example, we have two sub data groups **Sensitivity_Group#1** and **Sensitivity_Group#2** for this
specific hdf5 output for sensitivity analysis. Within these sub data groups, taking Sensitivity_Group#1 as an example, the output data is organized as shown in Figure 205.11.

![Figure 205.11: Datasets in Sensitivity_Group#1.](image)

It includes the following datasets:

* **Component_Sobol_Indexes_Sort** dataset:
  Is a column vector containing the computed Sobol Indexes for part of polynomial chaos (PC) bases specified through the sensitivity dimension group, in descending order.

* **PC_Sort** dataset:
  Is a 2D array that describes the sorted, part of PC bases specified through the sensitivity dimension group, corresponding to **Component_Sobol_Indexes_Sort** dataset. 
  \(PC_{Sort}{ij}\) denotes the order of polynomial chaos dimension \(\xi_j\) that contributes to the \(i^{th}\) sorted, PC basis.

* **Total_Sobol_Index** dataset:
  Is a scalar, total Sobol sensitivity index for PC bases specified in the sensitivity dimension group.

- **Generalized_Displacements_Sensitivity** data group:
  Contains Sobol sensitivity analysis results for stochastic nodal displacement response. The configuration of data structure for **Generalized_Displacements_Sensitivity** data group is the same as **Generalized_Accelerations_Sensitivity** data group.

- **Index_to_Global_Dimension** dataset:
  Contains a column vector of integers, which specifies the global dimension IDs for the polynomial chaos bases used to represent the nodal stochastic response.

- **PC** dataset:
  Is a 2D array that describes the multi-dimensional Hermite polynomial chaos (PC) bases for representing the nodal stochastic response. \(PC_{ij}\) denotes the order of polynomial chaos dimension
\( \xi_j \) that contributes to the \( i^{th} \) multidimensional Hermite PC basis.

- **PC_Variance** dataset:
  
  Is a column vector specifying the variances of Hermite PC basis.

- **Sensitivity_Dimension_Groups** data group:

  Contains the information about the sensitivity dimension groups defined for sensitivity analysis. Each sensitivity dimension group would be defined by a dataset within the data group as shown in Figure 205.12. Each dataset contains a column vector of integers, specifying the dimension IDs in specific sensitivity dimension group.

  ![Figure 205.12: Datasets in Sensitivity_Dimension_Groups data group.](image)

  Figure 205.12: Datasets in **Sensitivity_Dimension_Groups** data group.
205.3.6 Output Options

Real-ESSI Simulator outputs total displacements at all the nodes, as well total stress, total strain and total plastic strain at all the Gauss points of the element in each time step of each stage of loading. Real-ESSI also outputs any/all other element output in addition to the integration/Gauss point output. Generally, 3-D elements have only integration/Gauss point outputs and structure elements have only element output. The output options are reset to the default options in the beginning of each loading stage. More information about output organization is given in section 206.2.
205.3.6.1 Output Options: Enable/Disable Output

This option is used to enable or disable the outputting of results from all nodes and elements to HDF5 (.feioutput) output file.

**Note:** By default output is always enabled for each loading stage.

Command to disable output is

```
disable all output;
```

Command to enable output is

```
enable all output;
```
205.3.6.2 Output Options: Enable/Disable Element Output

This option is used to enable or disable the outputting of element results from all elements to HDF5 (.feioutput) output file, per stage of loading.

**Note:** By default all results from elements are output for each loading stage, so this option can be used to enable or disable output per loading stage.

Command to disable element output is

```
1 disable element output;
```

Command to enable element output is

```
1 enable element output;
```
205.3.6.3 Output Options: Enable/Disable Displacement Output

This option is used to enable or disable the displacement output at nodes to HDF5 (.feioutput) file. 

**Note:** By default displacement output is enabled.

Command to disable displacement output is

```
1 disable displacement output;
```

Command to enable displacement output is

```
1 enable displacement output;
```
205.3.6.4 Output Options: Enable/Disable Acceleration Output

This option is used to enable or disable the acceleration output at nodes to HDF5 (.feioutput) file.

**Note:** By default acceleration output is disabled.

Command to disable acceleration output is

```
1 disable acceleration output;
```

Command to enable acceleration output is

```
1 enable acceleration output;
```
205.3.6.5 Output Options: Enable/Disable Asynchronous Output

This option is used to enable or disable the asynchronous method of writing output to HDF5 (.feioutput) file.

Note: By default asynchronous output is disabled. Asynchronous output is an advanced output feature. Asynchronous output is suitable for I/O-bound simulation.

Command to disable asynchronous output is

```plaintext
1 disable asynchronous output;
```

Command to enable asynchronous output is

```plaintext
1 enable asynchronous output;
```
205.3.6.6 Output Options: Output Every n Steps

This option is used to output results at intervals of \( n \) time steps.

**Note:** By default results are output for every time step.

Command to enable output only at \( n^{th} \) time step interval

```
1 output every <.> steps;
```

For example: To output only at interval of two time steps for a simulation of 100 steps. One can write

```
1 output every 2 steps;
```

This will only output for steps 2,4,6,... until 100th step.
205.3.6.7 Output Options: Output Support Reactions

This option is used to output reactions at constrained supports.

**Note:** By default output reactions at constrained supports are disabled.

Command to enable reactions for support is

```plaintext
1 output support reactions;
```
205.4 Checking the Model

Real-ESSI provides model check capability:

```plaintext
check model;
```

This command will cycle over all the domain components, including Nodes, Elements, Loads, Constraints, etc. and execute the `checkModel()` function for each. Each domain component writes/reports to the terminal and to the `essi.log` file, if an error is found. For example, bricks will report when the computed Jacobian is negative and other similar errors. Nodes that are not connected will be reported as well. If the diagnostic log is empty, it means that the mesh has passed all tests. Additionally, an output HDF5 file is produced that can be used to display the mesh and do further visual inspections of the model. This file will have initial conditions as outputs for elements and nodes.

Command `check model;` represents a dry run through the model that is used to check the model before a full analysis. Model check is highly recommended before initial stages of a full analysis are executed.
205.5 Constitutive Testing

Material models can be tested using constitutive drivers which exercise single material models. RealE-SSI implements two such drivers.

1. Bardet Driver. Bardet-type constraints can be used to simulate conditions such as drained or undrained triaxial testing with strain or stress control or direct shear testing with shear control.

2. Direct Strain Driver. This driver applies a given strain history (specified by the user) to a material model.

Both these drivers produce identical output: the files Stress.feioutput and Strain.feioutput which contain stress and strain tensor components at each step. Additionally, the drivers may print out material internal information to the file Material.Output.feioutput. For the stress and strain files, each line of these files contain the stresses and strains organized in the following manner:

\[
\text{Stress.feioutput} \to \sigma_{11} \sigma_{22} \sigma_{33} \sigma_{12} \sigma_{13} \sigma_{23}.
\]
\[
\text{Strain.feioutput} \to \epsilon_{11} \epsilon_{22} \epsilon_{33} \epsilon_{12} \epsilon_{13} \epsilon_{23}.
\]

The Bardet driver has the following format.

```
simulate constitutive testing BARDETMETHOD use material # <.>
    scale_factor = <.>
    series_file = <string>
    sigma0 = ( <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> )
    verbose_output = <.>
```

Where,

- **BARDETMETHOD** can have any one of the following values:
  - **CONSTANT_P_TRIAXIAL_LOADING_STRAIN_CONTROL**: Triaxial loading with \( p \) kept constant. In this case the input file is interpreted as strain increments in the \( \epsilon_{11} \) component.
  - **DRAINED_TRIAXIAL_LOADING_STRESS_CONTROL**: Drained Triaxial loading. In this case the input file is interpreted as stress increments in the \( \sigma_{11} \) component.
  - **DRAINED_TRIAXIAL_LOADING_STRAIN_CONTROL**: Drained Triaxial loading. In this case the input file is interpreted as strain increments in the \( \epsilon_{11} \) component.
  - **UNDRAINED_TRIAXIAL_LOADING_STRAIN_CONTROL**: Undrained Triaxial loading. In this case the input file is interpreted as strain increments in the \( \epsilon_{11} \) component.
  - **UNDRAINED_TRIAXIAL_LOADING_STRESS_CONTROL**: Undrained Triaxial loading. In this case the input file is interpreted as stress increments in the \( \sigma_{11} \) component.
- **UNDRAINED_SIMPLE_SHEAR_LOADING_STRAIN_CONTROL**: Undrained simple-shear loading. In this case the input file is interpreted as angular strain increments in the $\gamma_{12} = 2\epsilon_{12}$ component.

- **scale_factor**: Can be used to scale the series file arbitrarily.

- **series_file**: String specifying the path to the file containing the increments (might be interpreted as strain or stress depending on the method chosen). Each line of the file contains one increment.

- **sigma0**: Components of the initial stress for the material, given in the order: $(\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{13}, \sigma_{23})$.

- **verbose_output (⇒N)** Whether the driver should print extra information about the material model every $N$ steps. If takes value 0 (no output) or $N$ (do output every $N$ increments). Each material implements its own output, so the format of the `Material_Output.feiooutput` file is variable and material dependent.

The direct strain driver has the following format.

```
simulate constitutive testing DIRECT_STRAIN use material # <.>
scale_factor = <.>
series_file = <string>
sigma0 = (<F/L^2>, <F/L^2>, <F/L^2>, <F/L^2>, <F/L^2>)
verbose_output = <.>
```

Where all the arguments are the same as the Bardet driver. In this case each line of the file contains all six components of the strain increment to be applied. For example:

```
series_file = "increments.txt" where each line in increments.txt contains $d\epsilon_{11}$ $d\epsilon_{22}$ $d\epsilon_{33}$ $d\epsilon_{12}$ $d\epsilon_{13}$ $d\epsilon_{23}$.
```
### 205.6 List of Available Commands (tentative, not up to date)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add acceleration field # &lt;.&gt; ax = &lt;accel&gt; ay = &lt;accel&gt; az = &lt;accel&gt; ;</td>
<td>Add an acceleration field with accelerations ax, ay, and az.</td>
</tr>
<tr>
<td>add constraint equal_dof with master node # &lt;.&gt; and slave node # &lt;.&gt;</td>
<td>Add a constraint that equalizes degrees of freedom between master and slave nodes.</td>
</tr>
<tr>
<td>add constraint equal_dof with node # &lt;.&gt; dof &lt;.&gt; master and node # &lt;.&gt;</td>
<td>Add a constraint that equalizes degree of freedom between master and slave.</td>
</tr>
<tr>
<td>add damping # &lt;.&gt; to element # &lt;.&gt;; add damping # &lt;.&gt; to node # &lt;.&gt;;</td>
<td>Add damping to an element or node.</td>
</tr>
<tr>
<td>add damping # &lt;.&gt; type Caughey3rd with a0 = &lt;1/time&gt; a1 = &lt;time&gt; a2 =</td>
<td>Add Caughey damping of order 3 with coefficients a0, a1, and a2.</td>
</tr>
<tr>
<td>&lt;time^3&gt; stiffness_to_use = &lt;Initial_Stiffness</td>
<td>Current_Stiffness</td>
</tr>
<tr>
<td>add damping # &lt;.&gt; type Caughey4th with a0 = &lt;1/time&gt; a1 = &lt;time&gt; a2 =</td>
<td>Add Caughey damping of order 4 with coefficients a0, a1, a2, and a3.</td>
</tr>
<tr>
<td>&lt;time^3&gt; a3 = &lt;time^5&gt; stiffness_to_use = &lt;Initial_Stiffness</td>
<td>Current_Stiffness</td>
</tr>
<tr>
<td>add damping # &lt;.&gt; type Rayleigh with a0 = &lt;1/time&gt; a1 = &lt;time&gt; stiffness_to_use</td>
<td>Add Rayleigh damping with coefficients a0 and a1 and stiffness use.</td>
</tr>
<tr>
<td>add domain reduction method loading # &lt;.&gt; hdf5_file = &lt;string&gt; scale_factor = &lt;.&gt;;</td>
<td>Add domain reduction method loading from HDF5 file with scale factor.</td>
</tr>
<tr>
<td>add element # &lt;.&gt; type 20NodeBrick using &lt;.&gt; Gauss points each direction</td>
<td>Add a 20-node brick element using Gauss points for each direction.</td>
</tr>
<tr>
<td>with nodes (&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>Use material # &lt;.&gt; for the element.</td>
</tr>
<tr>
<td>&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt; use material # &lt;.&gt;;</td>
<td></td>
</tr>
<tr>
<td>add element # &lt;.&gt; type 20NodeBrick with nodes (&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>Add a 20-node brick element with material # &lt;.&gt; and porosity.</td>
</tr>
<tr>
<td>&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>Use material # &lt;.&gt; and material properties.</td>
</tr>
<tr>
<td>&lt;.&gt; use material # &lt;.&gt; and porosity = &lt;.&gt; alpha = &lt;.&gt; rho_s = &lt;M/L^3&gt;</td>
<td></td>
</tr>
<tr>
<td>rho_f = &lt;M/L^3&gt; k_x = &lt;L^3T/M&gt; k_y = &lt;L^3T/M&gt; k_z = &lt;L^3T/M&gt; K_s =</td>
<td></td>
</tr>
<tr>
<td>K_f = &lt;stress&gt;;</td>
<td></td>
</tr>
<tr>
<td>add element # &lt;.&gt; type 20NodeBrick_up using &lt;.&gt; Gauss points each direction</td>
<td>Add an up element with Gauss points for each direction.</td>
</tr>
<tr>
<td>with nodes (&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>Use material # &lt;.&gt; and porosity = &lt;.&gt; alpha = &lt;.&gt; rho_s = &lt;M/L^3&gt; rho_f =</td>
</tr>
<tr>
<td>&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>M/L^3&gt; k_x = &lt;L^3T/M&gt; k_y = &lt;L^3T/M&gt; k_z = &lt;L^3T/M&gt; K_s = &lt;stress&gt; K_f =</td>
</tr>
<tr>
<td>&lt;stress&gt;;</td>
<td></td>
</tr>
<tr>
<td>add element # &lt;.&gt; type 20NodeBrick_up with nodes (&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>Add an up element with material # &lt;.&gt; and porosity.</td>
</tr>
<tr>
<td>&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>Use material # &lt;.&gt; and porosity = &lt;.&gt; alpha = &lt;.&gt; rho_s = &lt;M/L^3&gt; rho_f =</td>
</tr>
<tr>
<td>&lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;, &lt;.&gt;,</td>
<td>M/L^3&gt; k_x = &lt;L^3T/M&gt; k_y = &lt;L^3T/M&gt; k_z = &lt;L^3T/M&gt; K_s = &lt;stress&gt; K_f =</td>
</tr>
<tr>
<td>&lt;stress&gt;;</td>
<td></td>
</tr>
</tbody>
</table>
<N/L^3> k_x = <L^-3T/M> k_y = <L^-3T/M> k_z = <L^-3T/M> K_s = <stress> K_f = ← <stress>;
17 add element # .> type 27NodeBrick using .> Gauss points each direction with ← nodes (.>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .> use ← material # .>;
18 add element # .> type 27NodeBrick with nodes (.>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .> use ← material # .>;
19 add element # .> type 27NodeBrick_up using .> Gauss points each direction ← with nodes (.>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .> use ← material # .> and porosity = .> alpha = .> rho_s = <M/L^3> rho_f = ← <M/L^3> k_x = <L^-3T/M> k_y = <L^-3T/M> k_z = <L^-3T/M> K_s = <stress> K_f = ← <stress>;
20 add element # .> type 27NodeBrick_up with nodes (.>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .> use ← material # .> and porosity = .> alpha = .> rho_s = <M/L^3> rho_f = ← <M/L^3> k_x = <L^-3T/M> k_y = <L^-3T/M> k_z = <L^-3T/M> K_s = <stress> K_f = ← <stress>;
21 add element # .> type 27NodeBrick_upU using .> Gauss points each direction ← with nodes (.>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .> use ← material # .> and porosity = .> alpha = .> rho_s = <M/L^3> rho_f = ← <M/L^3> k_x = <L^-3T/M> k_y = <L^-3T/M> k_z = <L^-3T/M> K_s = <stress> K_f = ← <stress>;
22 add element # .> type 27NodeBrick_upU with nodes (.>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .> use ← material # .> and porosity = .> alpha = .> rho_s = <M/L^3> rho_f = ← <M/L^3> k_x = <L^-3T/M> k_y = <L^-3T/M> k_z = <L^-3T/M> K_s = <stress> K_f = ← <stress>;
23 add element # .> type 3NodeShell_ANDES with nodes (.>, .>, .>) use material ← # .> thickness = <l> ;
24 add element # .> type 4NodeShell_ANDES with nodes (.>, .>, .>, .>) use ← material # .> thickness = <l> ;
25 add element # .> type 4NodeShell_MITC4 with nodes (.>, .>, .>) use ← material # .> thickness = <L> ;
26 add element # .> type 4NodeShell_NewMITC4 with nodes (.>, .>, .>) use ← material # .> thickness = <L> ;
27 add element # .> type 8_27_NodeBrick using .> Gauss points each direction ← with nodes (.>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .>, .> use ← material # .>;
add element # <.> type 8_27_NodeBrick with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>:

add element # <.> type 8_27_NodeBrick_up using <.> Gauss points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> and porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>:

add element # <.> type 8_27_NodeBrick_up with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> and porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>:

add element # <.> type 8_27_NodeBrick_upU using <.> Gauss points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> and porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>:

add element # <.> type 8NodeBrick using <.> Gauss points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>:

add element # <.> type 8NodeBrick with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>:

add element # <.> type 8NodeBrick_up using <.> Gauss points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>:

add element # <.> type 8NodeBrick_up with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>:

add element # <.> type 8NodeBrick_upU using <.> Gauss points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>:

add element # <.> type 8NodeBrick_upU with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>:
K_f = <stress>;  
39 add element # <.> type beam_9dof_elastic with nodes (<.>, <.>) cross_section = ←
   <area> elastic_modulus = <F/L^2> shear_modulus = <F/L^2> torsion_Jx = ←
   <length^4> bending_Iy = <length^4> bending_Iz = <length^4> mass_density = ←
   <M/L^3> xz_plane_vector = (<.>, <.>, <.> ) joint_1_offset = (<L>, <L>, <L> ←
   ) joint_2_offset = (<L>, <L>, <L> );
40 add element # <.> type beam_displacement_based with nodes (<.>, <.>) with # <.> ←
   integration_points use section # <.> mass_density = <M/L^3> IntegrationRule ←
   = "" xz_plane_vector = (<.>, <.>, <.> ) joint_1_offset = (<L>, <L>, <L> ) ←
   joint_2_offset = (<L>, <L>, <L> );
41 add element # <.> type beam_elastic with nodes (<.>, <.>) cross_section = ←
   <area> elastic_modulus = <F/L^2> shear_modulus = <F/L^2> torsion_Jx = ←
   <length^4> bending_Iy = <length^4> bending_Iz = <length^4> mass_density = ←
   <M/L^3> xz_plane_vector = (<.>, <.>, <.> ) joint_1_offset = (<L>, <L>, <L> ←
   ) joint_2_offset = (<L>, <L>, <L> );
42 add element # <.> type beam_elastic_lumped_mass with nodes (<.>, <.>) ←
   cross_section = <area> elastic_modulus = <F/L^2> shear_modulus = <F/L^2> ←
   torsion_Jx = <length^4> bending_Iy = <length^4> bending_Iz = <length^4> mass_density = ←
   <M/L^3> xz_plane_vector = (<.>, <.>, <.> ) joint_1_offset = (<L>, <L>, <L> ←
   ) joint_2_offset = (<L>, <L>, <L> );
43 add element # <.> type BeamColumnDispFiber3d with nodes (<.>, <.>) ←
   number_of_integration_points = <.> section_number = <.> mass_density = ←
   <M/L^3> xz_plane_vector = (<.>, <.>, <.> ) joint_1_offset = (<L>, <L>, <L> ←
   ) joint_2_offset = (<L>, <L>, <L> );
44 add element # <.> type HardContact with nodes (<.>, <.>) axial_stiffness = ←
   <F/L> shear_stiffness = <F/L> normal_damping = <F/L> tangential_damping = ←
   <F/L> friction_ratio = <.> contact_plane_vector = (<.>, <.>, <.> );
45 add element # <.> type HardWetContact with nodes (<.>, <.>) axial_stiffness = ←
   <F/L> shear_stiffness = <F/L> normal_damping = <F/L> tangential_damping = ←
   <F/L> friction_ratio = <.> contact_plane_vector = (<.>, <.>, <.> );
46 add element # <.> type SoftContact with nodes (<.>, <.>) use material # <.> ←
   initial_axial_stiffness = <F/L> stiffening_rate = <m^-1> shear_stiffness = ←
   <F/L> normal_damping = <F/L> tangential_damping = <F/L> friction_ratio = ←
   <.> contact_plane_vector = (<.>, <.>, <.> );
47 add element # <.> type SoftWetContact with nodes (<.>, <.>) ←
   initial_axial_stiffness = <F/L> stiffening_rate = <m^-1> shear_stiffness = ←
   <F/L> normal_damping = <F/L> tangential_damping = <F/L> friction_ratio = ←
   <.> contact_plane_vector = (<.>, <.>, <.> );
48 add element # <.> type truss with nodes (<.>, <.>) use material # <.> ←
   cross_section = <length^2> mass_density = <M/L^3> ;
49 add element # <.> type variable_node_brick_8_to_27 using <.> Gauss points each ←
51 add elements (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>) use material # <.>;  
52 add fiber # <.> using material # <.> to section # <.> fiber_cross_section = ↩  
<area> fiber_location = (<L>,<L>);  
53 add imposed motion # <.> to node # <.> dof DOFTYPE displacement_scale_unit =  
<displacement> displacement_file = "disp_filename" velocity_scale_unit =  
<velocity> velocity_file = "vel_filename" acceleration_scale_unit =  
<acceleration> acceleration_file = "acc_filename";  
54 add imposed motion # <.> to node # <.> dof DOFTYPE time_step = <t>  
displacement_scale_unit = <length> displacement_file = "disp_filename"  
velocity_scale_unit = <velocity> velocity_file = "vel_filename"  
acceleration_scale_unit = <acceleration> acceleration_file = "acc_filename";  
55 add load # <.> to all elements type self_weight use acceleration field # <.>;  
56 add load # <.> to element # <.> type self_weight use acceleration field # <.>;  
57 add load # <.> to element # <.> type surface at nodes (<.>,<.>,<.>,<.>) with magnitude <.>;  
58 add load # <.> to element # <.> type surface at nodes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>) with magnitudes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>);  
59 add load # <.> to element # <.> type surface at nodes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>) with magnitude <.>;  
60 add load # <.> to element # <.> type surface at nodes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>) with magnitudes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>);  
61 add load # <.> to element # <.> type surface at nodes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>) with magnitude <.>;  
62 add load # <.> to element # <.> type surface at nodes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>) with magnitudes (<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>,<.>);  
63 add load # <.> to node # <.> type from_reactions;  
64 add load # <.> to node # <.> type linear FORCETYPE = <force or moment>;  
FORCETYPE = Fx Fy Fz Mx Mz F_fluid_x F_fluid_y F_fluid_z  
65 add load # <.> to node # <.> type path_series FORCETYPE = <force or moment>  
time_step = <time> series_file = "filename";  
66 add load # <.> to node # <.> type path_time_series FORCETYPE = <force or moment>  
series_file = "filename";  
67 add load # <.> to node # <.> type self_weight use acceleration field # <.>;  
68 add mass to node # <.> mx = <mass> my = <mass> mz = <mass> Imx =  
<mass*length^2> Imy = <mass*length^2> Imz = <mass*length^2>;  
69 add mass to node # <.> mx = <mass> my = <mass> mz = <mass>;  
70 add material # <.> type CamClay mass_density = <M/L^3> M = <.> lambda = <.>  
kappa = <.> e0 = <.> p0 = <F/L^2> Poisson_ratio = <.>  
initial_confining_stress = <F/L^2>
add material # <.> type DruckerPrager mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.> druckerprager_k = <.> kinematic_hardening_rate = <.> isotropic_hardening_rate = \( <F/L^2> \) initial_confining_stress = exp;
add material # <.> type DruckerPragerArmstrongFrederickLE mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.> druckerprager_k = <.> armstrong_frederick_ha = \( <F/L^2> \) armstrong_frederick_cr = \( <F/L^2> \) isotropic_hardening_rate = \( <F/L^2> \) initial_confining_stress = \( <F/L^2> \);
add material # <.> type DruckerPragerArmstrongFrederickNE mass_density = \( <M/L^3> \) DuncanChang_K = <.> DuncanChang_pa = \( <F/L^2> \) DuncanChang_n = <.> DuncanChang_sigma3_max = \( <F/L^2> \) DuncanChang_nu = <.> armstrong_frederick_ha = \( <F/L^2> \) armstrong_frederick_cr = \( <F/L^2> \) isotropic_hardening_rate = \( <F/L^2> \) initial_confining_stress = \( <F/L^2> \);
add material # <.> type DruckerPragerNonAssociateArmstrongFrederick mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.> druckerprager_k = <.> armstrong_frederick_ha = \( <F/L^2> \) armstrong_frederick_cr = \( <F/L^2> \) isotropic_hardening_rate = \( <F/L^2> \) initial_confining_stress = \( <F/L^2> \) plastic_flow_xi = <> plastic_flow_kd = <>;
add material # <.> type DruckerPragerNonAssociateLinearHardening mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.> druckerprager_k = <.> kinematic_hardening_rate = \( <F/L^2> \) isotropic_hardening_rate = \( <F/L^2> \) initial_confining_stress = \( <F/L^2> \) plastic_flow_xi = <> plastic_flow_kd = <>;
add material # <.> type DruckerPragervonMises mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.> druckerprager_k = <.> kinematic_hardening_rate = \( <F/L^2> \) isotropic_hardening_rate = \( <F/L^2> \) initial_confining_stress = exp;
add material # <.> type linear_elastic_crossanisotropic mass_density = \( <mass\> \) elastic_modulus_horizontal = \( <F/L^2> \) elastic_modulus_vertical = \( <F/L^2> \) poisson_ratio_h_v = <.> shear_modulus_h_v = \( <F/L^2> \);
add material # <.> type linear_elastic_isotropic_3d mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.>
add material # <.> type linear_elastic_isotropic_3d_LT mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.>
add material # <.> type roundedMohrCoulomb mass_density = \( <M/L^3> \) elastic_modulus = \( <F/L^2> \) poisson_ratio = <.> RMC_m = <.> RMC_qa = \( <F/L^2> \) RMC_pc = \( <F/L^2> \) RMC_e = <.> RMC_eta0 = <.> RMC_Heta = \( <F/L^2> \) initial_confining_stress = \( <F/L^2> \)
Real-ESSI Lecture Notes

205.6. LIST OF AVAILABLE COMMANDS (TEN... page: 1115 of 3091

sanisand2004_nb = <.> sanisand2004_A0 = <.> sanisand2004_nd = <.> ↔
sanisand2004_z_max = <.> sanisand2004_cz = <.> initial_confining_stress = ↔
<stress> ;
82 add material # <.> type sanisand2004_legacy mass_density = <M/L^3> e0 = <.> ↔
sanisand2004_G0 = <.> poisson_ratio = <.> sanisand2004_Pat = <stress> ↔
sanisand2004_lambda_c = <.> sanisand2004_xi = <.> sanisand2004_ec_ref = <.> ↔
sanisand2004_m = <.> sanisand2004_h0 = <.> sanisand2004_ch = <.> ↔
sanisand2004_nb = <.> sanisand2004_A0 = <.> sanisand2004_nd = <.> ↔
sanisand2004_z_max = <.> sanisand2004_cz = <.> initial_confining_stress = ↔
<stress> algorithm = <explicit|implicit> number_of_subincrements = <.> ↔
maximum_number_of_iterations = <.> tolerance_1 = <.> tolerance_2 = <.>;
83 add material # <.> type sanisand2008 mass_density = <M/L^3> e0 = <.> ↔
sanisand2008_G0 = <.> sanisand2008_K0 = <.> sanisand2008_Pat = <stress> ↔
sanisand2008_k_c = <.> sanisand2008_alpha_cc = <.> sanisand2008_c = <.> ↔
sanisand2008_xi = <.> sanisand2008_lambda = <.> sanisand2008_ec_ref = <.> ↔
sanisand2008_m = <.> sanisand2008_h0 = <.> sanisand2008_ch = <.> ↔
sanisand2008_nb = <.> sanisand2008_A0 = <.> sanisand2008_nd = <.> ↔
sanisand2008_p_r = <.> sanisand2008_rho_c = <.> sanisand2008_theta_c = <.> ↔
sanisand2008_p0 = <stress> sanisand2008_p_in = <.> algorithm = ↔
<explicit|implicit> number_of_subincrements = <.> ↔
maximum_number_of_iterations = <.> tolerance_1 = <.> tolerance_2 = <.>;
84 add material # <.> type uniaxial_concrete02 compressive_strength = <F/L^2> ↔
strain_at_compressive_strength = <.> crushing_strength = <F/L^2> ↔
strain_at_crushing_strength = <.> lambda = <.> tensile_strength = <F/L^2> ↔
tension_softening_stiffness = <F/L^2>; 85 add material # <.> type uniaxial_elastic elastic_modulus = <F/L^2> ↔
viscoelastic_modulus = <mass / length / time> ;
86 add material # <.> type uniaxial_steel01 yield_strength = <F/L^2> ↔
elastic_modulus = <F/L^2> strain_hardening_ratio = <.> a1 = <.> a2 = <.> a3 ↔
= <> a4 = <.> ;
87 add material # <.> type uniaxial_steel02 yield_strength = <F/L^2> ↔
elastic_modulus = <F/L^2> strain_hardening_ratio = <.> R0 = <.> cR1 = <.> ↔
cR2 = <> a1 = <> a2 = <> a3 = <> a4 = <> ;
88 add material # <.> type vonMises mass_density = <M/L^3> elastic_modulus = ↔
<F/L^2> poisson_ratio = <.> von_mises_radius = <F/L^2> ↔
kinematic_hardening_rate = <F/L^2> isotropic_hardening_rate = <F/L^2> ;
89 add material # <.> type vonMisesArmstrongFrederick mass_density = <M/L^3> ↔
elastic_modulus = <F/L^2> poisson_ratio = <.> von_mises_radius = <> ↔
armstrong_frederick_ha = <F/L^2> armstrong_frederick_cr = <F/L^2> ↔
isotropic_hardening_rate = <F/L^2> ;
90 add node # <.> at (<length>,<length>,<length>) with <.> dofs;
91 add nodes (<.>) to physical_node_group "string";
add section # <.> type elastic3d elastic_modulus = <F/L^2> cross_section = <L^2> bending_Iz = <L^4> bending_Iy=<L^4> torsion_Jx=<L^4> ;
add section # <.> type Elastic_Membrane_Plate elastic_modulus = <F/L^2> poisson_ratio = <.> thickness = <length> mass_density = <M/L^3>;
add section # <.> type FiberSection TorsionConstant_GJ = <F*L^2>
add section # <.> type Membrane_Plate_Fiber thickness = <length> use material # <.>;
add single point constraint to node # <.> dof to constrain <dof_type> constraint value of <corresponding unit>;
add uniform acceleration # <.> to all nodes dof <.> time_step = <T> scale_factor = <.> initial_velocity = <L/S> acceleration_file = <string>;
check mesh filename;
compute reaction forces;
define algorithm With_no_convergence_check / Newton / Modified_Newton;
define convergence test Norm_Displacement_Increment / Energy_Increment / Norm_Unbalance / Relative_Norm_Displacement_Increment / Relative_Energy_Increment / Relative_Norm_Unbalance tolerance = <.> maximum_iterations = <.> verbose_level = <0>|<1>|<2>;
define dynamic integrator Hilber_Hughes_Taylor with alpha = <.>;
define dynamic integrator Newmark with gamma = <.> beta = <.>;
define load factor increment <.>;
define NDMaterial constitutive integration algorithm Forward_Euler;
define NDMaterial constitutive integration algorithm Forward_Euler_Subincrement number_of_subincrements =<.>;
define NDMaterial constitutive integration algorithm Forward_Euler|Forward_Euler_Subincrement|Backward_Euler|Backward_Euler_Subincrement| yield_function_relative_tolerance = <.> stress_relative_tolerance = <.> maximum_iterations = <.>;
define physical_element_group "string";
define physical_node_group "string";
define solver ProfileSPD / UMFPack;
define static integrator displacement_control using node # <.> dof DOFTYPE increment <length>;
disable asynchronous output;
disable element output;
disable output;
enable asynchronous output;
enable element output;
enable output;
fix node # <.> dofs <.>;
fix node # <.> dofs all;
free node # <.> dofs <.>;
help;
if (.) { } else {};}
if (. ) { 
model name "name_string";
new loading stage "name_string";
output every <.> steps;
output non_converged_iterations;
output support reactions;
print <.>;
print element # <.>;
print node # <.>;
print physical_element_group "string";
print physical_node_group "string";
remove constraint equal_dof node # <.>;
remove displacement from node # <.>;
remove element # <.>;
remove imposed motion # <.>;
remove load # <.>;
remove node # <.>;
remove physical_node_group "string";
remove strain from element # <.>;
remove physical_element_group "string";
runTest;
set output compression level to <.>;
simulate <.> steps using static algorithm;
simulate <.> steps using transient algorithm time_step = <time> ←
simulate <.> steps using variable transient algorithm time_step = <time> ←
    minimum_time_step = <time> maximum_time_step = <time> number_of_iterations ←
    = <.>;
simulate constitutive testing BARDETMETHOD use material # <.> scale_factor = ←
    <.> series_file = <string> sigma0 = ( <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> ←
    , <F/L^2> , <F/L^2> ) verbose_output = <.>
simulate constitutive testing constant mean pressure triaxial strain control ←
    use material # <.> strain_increment_size = <.> maximum_strain = <.> ←
    number_of_times_reaching_maximum_strain = <.>;
simulate constitutive testing DIRECT_STRAIN use material # <.> scale_factor = ←
    <.> series_file = <string> sigma0 = ( <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> ←
    , <F/L^2> , <F/L^2> ) verbose_output = <.>
simulate constitutive testing drained triaxial strain control use material # ←
<.> strain_increment_size = <.> maximum_strain = <.> ←
    number_of_times_reaching_maximum_strain = <.>;
simulate constitutive testing undrained simple shear use material # <.> ←
    strain_increment_size = <.> maximum_strain = <.> ←
    number_of_times_reaching_maximum_strain = <.>;
simulate constitutive testing undrained triaxial stress control use material # ←
<.> strain_increment_size = <.> maximum_strain = <.> ←
number_of_times_reaching_maximum_strain = <.>;
154 simulate constitutive testing undrained triaxial use material # <.> ←
  strain_increment_size = <.> maximum_strain = <.> ←
  number_of_times_reaching_maximum_strain = <.>;
155 simulate using eigen algorithm number_of_modes = <.>;
156 ux uy uz Ux Uy Uz rx ry rz;
157 while (. ) { };
158 whos;
205.7 List of reserved keywords

The following keywords are reserved and cannot be used as variables in a script or interactive session. Doing so would result in a syntax error.

First Order (commands)

```
a0
a1
a2
a3
a4
acceleration
acceleration_depth
acceleration_file
acceleration_filename
acceleration_scale_unit
add
algorithm
algorithm
all
all
allowed_subincrement_strain
alpha
alpha1
alpha2
and
angle
armstrong_frederick_cr
armstrong_frederick_ha
asynchronous
at
ax
axial_penalty_stiffness
axial_stiffness
axial_viscous_damping
ay
az
bending_Iy
bending_Iz
beta_min
case
cases
characteristic_strength
check
chi
cohesion
combine
compression
compressive_strength
compressive_yield_strength
```
compute
confinement
confinement_strain
constitutive
constrain
constraint
contact_plane_vector
control
convergence
cR1
cR2
cross_section
crushing_strength
Current_Stiffness
cyclic
damage_parameter_An
damage_parameter_Ap
damage_parameter_Bn
damping
define
depth
dilatancy_angle
dilation_angle_eta
dilation_scale
direction
disable
displacement
displacement_file
displacement_scale_unit
dof
dofs
dofs
domain
druckerprager_k
DuncanChang_K
DuncanChang_n
DuncanChang_nu
DuncanChang_pa
DuncanChang_sigma3_max
DYNAMIC_DOMAIN_PARTITION
e0
each
elastic
elastic_modulus
elastic_modulus_horizontal
elastic_modulus_vertical
element
elements
else
enable
every
factor
fiber
fiber_cross_section
fiber_location
field
file
fix
fluid
free
friction_angle
friction_ratio
from
Gauss
generate
GoverGmax
h_in
hardening_parameters_of_yield_surfaces
hardening_parameters_scale_unit
hdf5_file
hdf5_filenames_list
help
if
imposed
Imx
Imy
Imz
in
inclined
increment
initial_axial_stiffness
initial_confining_stress
initial_elastic_modulus
initial_shear_modulus
initial_shear_stiffness
initial_velocity
integration
integration_points
IntegrationRule
integrator
interface
isotropic_hardening_rate
joint_1_offset
joint_2_offset
K_f
K_s
k_x
k_y
k_z
kappa
kd_in
kinematic_hardening_rate
| 147 | lambda  |
| 148 | level   |
| 149 | line_search_beta |
| 150 | line_search_eta |
| 151 | line_search_max_iter |
| 152 | liquefaction_Alpha |
| 153 | liquefaction_c_h0 |
| 154 | liquefaction_Dir |
| 155 | liquefaction_dre1 |
| 156 | liquefaction_Dre2 |
| 157 | liquefaction_EXPN |
| 158 | liquefaction_gamar |
| 159 | liquefaction_mdc |
| 160 | liquefaction_mfc |
| 161 | liquefaction_pa |
| 162 | liquefaction_pmin |
| 163 | load    |
| 164 | load_factors_list |
| 165 | loading |
| 166 | local_y_vector |
| 167 | local_z_vector |
| 168 | M       |
| 169 | M_in    |
| 170 | magnitude |
| 171 | magnitudes |
| 172 | mass    |
| 173 | mass_density |
| 174 | master  |
| 175 | material |
| 176 | max_axial_stiffness |
| 177 | maximum_iterations |
| 178 | maximum_number_of_iterations |
| 179 | maximum_strain |
| 180 | maximum_stress |
| 181 | maximum_time_step |
| 182 | method  |
| 183 | minimal |
| 184 | minimum_time_step |
| 185 | model   |
| 186 | model   |
| 187 | moment_x_stiffness |
| 188 | moment_y_stiffness |
| 189 | monotonic |
| 190 | motion  |
| 191 | mu      |
| 192 | mx      |
| 193 | my      |
| 194 | mz      |
| 195 | name    |
| 196 | NDMaterial |
| 197 | new     |
newton_with_subincrement
node
nodes
number_of_cycles
number_of_files
number_of_increment
number_of_integration_points
number_of_iterations
number_of_layers
number_of_modes
number_of_subincrements
number_of_times_reaching_maximum_strain
of
output
output
output_filename
p0
parallel
peak_friction_coefficient_limit
peak_friction_coefficient_rate_of_decrease
penalty_stiffness
pi1
pi2
pi3
plastic_deformation_rate
plastic_flow_kd
plastic_flow_xi
plot
point
points
poisson_ratio
poisson_ratio_h_h
poisson_ratio_h_v
porosity
print
propagation
pure
R0
radiuses_of_yield_surface
radiuses_scale_unit
rate_of_softening
reduction
reference_pressure
remove
remvpe
residual_friction_coefficient
restart
restart_files
results
rho_a
rho_f
rho_s
rho_w
RMC_e
RMC_\eta_0
RMC_Heta
RMC_m
RMC_pc
RMC_qa
RMC_shape_k
rounded_distance
runTest
sanisand2004_A0
sanisand2004_c
sanisand2004_ch
sanisand2004_cz
sanisand2004_ec_ref
sanisand2004_G0
sanisand2004_h0
sanisand2004_lambda_c
sanisand2004_m
sanisand2004_Mc
sanisand2004_nb
sanisand2004_nd
sanisand2004_p\_cut
sanisand2004_Pat
sanisand2004_xi
sanisand2004_z\_max
sanisand2008_A0
sanisand2008_alpha_cc
sanisand2008_c
sanisand2008_ch
sanisand2008_cz
sanisand2008_ec_ref
sanisand2008_G0
sanisand2008_h0
sanisand2008_K0
sanisand2008_k\_c
sanisand2008_lambda
sanisand2008_m
sanisand2008_nb
sanisand2008_nd
sanisand2008_p0
sanisand2008_p\_in
sanisand2008_p\_r
sanisand2008_Pat
sanisand2008_rho\_c
sanisand2008_theta\_c
sanisand2008_X
sanisand2008_xi
sanisand2008_z\_max
save
scale_factor
SCOTCHGRAPHPARTITIONER
section
section_number
sequential
series_file
set
shear
shear_length_ratio
shear_modulus
shear_modulus_h_v
shear_stiffness
shear_viscous_damping
shear_zone_thickness
ShearStrainGamma
sigma0
simulate
single
size_of_peak_plateau
sizes_of_yield_surfaces
slave
slave
soil
soil_profile_filename
soil_surface
solid
solver
stage
steps
stiffening_rate
stiffness_to_use
strain
strain_at_compressive_strength
strain_at_crushing_strength
strain_hardening_ratio
strain_increment_size
stress
stress_increment_size
stress_relative_tolerance
sub-stepping
surface
surface_vector_relative_tolerance
tensile_strength
tensile_yield_strength
tension_softening_stiffness
test
test
testing
thickness
time_step
tolerance_1
tolerance_2
torsion_Jx
torsional_stiffness
TorsionConstant_GJ
total_number_of_shear_modulus
total_number_of_yield_surface
triaxial
type
uniaxial
uniaxial_material
uniform
unit_of_acceleration
unit_of_damping
unit_of_rho
unit_of_vs
use
using
value
velocity_file
velocity_scale_unit
verbose_output
viscoelastic_modulus
von_mises_radius
wave
wave1c
wave3c
while
whos
with
xi_in
xz_plane_vector
yield_function_relative_tolerance
yield_strength
yield_surface_scale_unit
x
y
z

Second Order (inside commands)

20NodeBrick
20NodeBrick_up
20NodeBrick_upU
27NodeBrick
27NodeBrick_up
27NodeBrick_upU
3NodeShell_ANDES
4NodeShell_ANDES
4NodeShell/MITC4

Jeremić et al., University of California, Davis
Absolute_Norm_Displacement_Increment
Absolute_Norm_Unbalanced_Force
arclength_control
Average_Norm_Displacement_Increment
Average_Norm_Unbalanced_Force
Backward_Euler
BARDETMETHOD
beam_9dof_elastic
beam_displacement_based
beam_elastic
beam_elastic_lumped_mass
BeamColumnDispFiber3d
BearingElastomericPlasticity3d
BFGS
BondedContact
CamClay
Caughey3rd
Caughey4th
canonical mean pressure triaxial strain control
Caughey8NodeBrick
Caughey4th
Cameron
cosserat8NodeBrick
Caughey8NodeBrick
Caughey8NodeBrick_up
Caughey8NodeBrick_upU
Caughey9th
Caughey9th
Caughey4th
constant mean pressure triaxial strain control
Cosserat8NodeBrick
Cosserat_linear_elastic_isotropic_3d
Cosserat_von_Mises
DIRECT_STRAIN
displacement_control
DOFTYPE
domain reduction method
drained triaxial strain control
DruckerPrager
DruckerPragerArmstrongFrederickLE
DruckerPragerArmstrongFrederickNE
DruckerPragerMultipleYieldSurface
DruckerPragerMultipleYieldSurfaceGoverGmax
DruckerPragerNonAssociateArmstrongFrederick
DruckerPragerNonAssociateLinearHardening
DruckerPragervonMises
dynamic
eigen
elastic3d
Elastic_Membrane_Plate
ElasticFourNodeQuad
Energy_Increment
equal_dof
F_fluid_x
61 F\_fluid\_y
62 F\_fluid\_z
63 FiberSection
64 ForceBasedCoupledHardContact
65 ForceBasedCoupledSoftContact
66 ForceBasedElasticContact
67 ForceBasedHardContact
68 ForceBasedSoftContact
69 FORCETYPE
70 Forward\_Euler
71 Forward\_Euler\_Subincrement
72 from\_reactions
73 Fx
74 Fy
75 Fz
76 Hilber\_Hughes\_Taylor
77 HyperbolicDruckerPragerArmstrongFrederick
78 HyperbolicDruckerPragerLinearHardening
79 HyperbolicDruckerPragerNonAssociateArmstrongFrederick
80 HyperbolicDruckerPragerNonAssociateLinearHardening
81 linear
82 linear\_elastic\_crossanisotropic
83 linear\_elastic\_isotropic\_3d
84 linear\_elastic\_isotropic\_3d\_LT
85 Membrane\_Plate\_Fiber
86 Modified\_Newton
87 Mx
88 My
89 Mz
90 Newmark
91 Newton
92 non\_converged\_iterations
93 NonlinearFourNodeQuad
94 Norm\_Displacement\_Increment
95 Norm\_Unbalance
96 Parallel
97 path\_series
98 path\_time\_series
99 petsc
100 petsc\_options\_string
101 physical\_element\_group
102 physical\_node\_group
103 Pisano
104 PlaneStressLayeredMaterial
105 PlaneStressRebarMaterial
106 PlasticDamageConcretePlaneStress
107 pressure
108 ProfileSPD
109 Rayleigh
110 reaction forces
111 reactions
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td>Relative_Energy_Increment</td>
</tr>
<tr>
<td>113</td>
<td>Relative_Norm_Displacement_Increment</td>
</tr>
<tr>
<td>114</td>
<td>Relative_Norm_Unbalance</td>
</tr>
<tr>
<td>115</td>
<td>Relative_Norm_Unbalanced_Force</td>
</tr>
<tr>
<td>116</td>
<td>roundedMohrCoulomb</td>
</tr>
<tr>
<td>117</td>
<td>RoundedMohrCoulombMultipleYieldSurface</td>
</tr>
<tr>
<td>118</td>
<td>sanisand2004</td>
</tr>
<tr>
<td>119</td>
<td>sanisand2004_legacy</td>
</tr>
<tr>
<td>120</td>
<td>sanisand2008</td>
</tr>
<tr>
<td>121</td>
<td>self_weight</td>
</tr>
<tr>
<td>122</td>
<td>ShearBeam</td>
</tr>
<tr>
<td>123</td>
<td>solid_fluid_interaction_transient</td>
</tr>
<tr>
<td>124</td>
<td>static</td>
</tr>
<tr>
<td>125</td>
<td>StressBasedCoupledHardContact_E1PPlShear</td>
</tr>
<tr>
<td>126</td>
<td>StressBasedCoupledHardContact_NonLinHardShear</td>
</tr>
<tr>
<td>127</td>
<td>StressBasedCoupledHardContact_NonLinHardSoftShear</td>
</tr>
<tr>
<td>128</td>
<td>StressBasedCoupledSoftContact</td>
</tr>
<tr>
<td>129</td>
<td>StressBasedCoupledSoftContact_E1PPlShear</td>
</tr>
<tr>
<td>130</td>
<td>StressBasedCoupledSoftContact_NonLinHardShear</td>
</tr>
<tr>
<td>131</td>
<td>StressBasedCoupledSoftContact_NonLinHardSoftShear</td>
</tr>
<tr>
<td>132</td>
<td>StressBasedHardContact_E1PPlShear</td>
</tr>
<tr>
<td>133</td>
<td>StressBasedHardContact_NonLinHardShear</td>
</tr>
<tr>
<td>134</td>
<td>StressBasedHardContact_NonLinHardSoftShear</td>
</tr>
<tr>
<td>135</td>
<td>StressBasedSoftContact_E1PPlShear</td>
</tr>
<tr>
<td>136</td>
<td>StressBasedSoftContact_NonLinHardShear</td>
</tr>
<tr>
<td>137</td>
<td>StressBasedSoftContact_NonLinHardSoftShear</td>
</tr>
<tr>
<td>138</td>
<td>SuperElementLinearElasticImport</td>
</tr>
<tr>
<td>139</td>
<td>support</td>
</tr>
<tr>
<td>140</td>
<td>surface</td>
</tr>
<tr>
<td>141</td>
<td>transient</td>
</tr>
<tr>
<td>142</td>
<td>truss</td>
</tr>
<tr>
<td>143</td>
<td>TsinghuaLiquefactionModelCirclePiPlane</td>
</tr>
<tr>
<td>144</td>
<td>TsinghuaLiquefactionModelNonCirclePiPlane</td>
</tr>
<tr>
<td>145</td>
<td>UMFPack</td>
</tr>
<tr>
<td>146</td>
<td>undrained_simple_shear</td>
</tr>
<tr>
<td>147</td>
<td>undrained_triaxial</td>
</tr>
<tr>
<td>148</td>
<td>undrained_triaxial_stress_control</td>
</tr>
<tr>
<td>149</td>
<td>uniaxial_concrete02</td>
</tr>
<tr>
<td>150</td>
<td>uniaxial_elastic</td>
</tr>
<tr>
<td>151</td>
<td>uniaxial_steel01</td>
</tr>
<tr>
<td>152</td>
<td>uniaxial_steel02</td>
</tr>
<tr>
<td>153</td>
<td>variable_transient</td>
</tr>
<tr>
<td>154</td>
<td>variable_node_brick_8_to_27</td>
</tr>
<tr>
<td>155</td>
<td>vonMises</td>
</tr>
<tr>
<td>156</td>
<td>vonMisesArmstrongFrederick</td>
</tr>
<tr>
<td>157</td>
<td>vonMisesMultipleYieldSurface</td>
</tr>
<tr>
<td>158</td>
<td>vonMisesMultipleYieldSurfaceGoverGmax</td>
</tr>
<tr>
<td>159</td>
<td>With_no_convergence_check</td>
</tr>
<tr>
<td>160</td>
<td></td>
</tr>
<tr>
<td>161</td>
<td>beta</td>
</tr>
<tr>
<td>162</td>
<td>gamma</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>163</td>
<td>delta</td>
</tr>
<tr>
<td>164</td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>ux</td>
</tr>
<tr>
<td>166</td>
<td>uy</td>
</tr>
<tr>
<td>167</td>
<td>uz</td>
</tr>
<tr>
<td>168</td>
<td>rx</td>
</tr>
<tr>
<td>169</td>
<td>ry</td>
</tr>
<tr>
<td>170</td>
<td>rz</td>
</tr>
<tr>
<td>171</td>
<td>Ux</td>
</tr>
<tr>
<td>172</td>
<td>Uy</td>
</tr>
<tr>
<td>173</td>
<td>Uz</td>
</tr>
<tr>
<td>174</td>
<td>p</td>
</tr>
<tr>
<td>175</td>
<td>M</td>
</tr>
<tr>
<td>176</td>
<td>m</td>
</tr>
<tr>
<td>177</td>
<td>kg</td>
</tr>
<tr>
<td>178</td>
<td>s</td>
</tr>
<tr>
<td>179</td>
<td>cm</td>
</tr>
<tr>
<td>180</td>
<td>mm</td>
</tr>
<tr>
<td>181</td>
<td>km</td>
</tr>
<tr>
<td>182</td>
<td>Hz</td>
</tr>
<tr>
<td>183</td>
<td>Minute</td>
</tr>
<tr>
<td>184</td>
<td>Hour</td>
</tr>
<tr>
<td>185</td>
<td>Day</td>
</tr>
<tr>
<td>186</td>
<td>Week</td>
</tr>
<tr>
<td>187</td>
<td>ms</td>
</tr>
<tr>
<td>188</td>
<td>ns</td>
</tr>
<tr>
<td>189</td>
<td>N</td>
</tr>
<tr>
<td>190</td>
<td>kN</td>
</tr>
<tr>
<td>191</td>
<td>Pa</td>
</tr>
<tr>
<td>192</td>
<td>kPa</td>
</tr>
<tr>
<td>193</td>
<td>MPa</td>
</tr>
<tr>
<td>194</td>
<td>GPa</td>
</tr>
<tr>
<td>195</td>
<td>pound</td>
</tr>
<tr>
<td>196</td>
<td>lbm</td>
</tr>
<tr>
<td>197</td>
<td>lbf</td>
</tr>
<tr>
<td>198</td>
<td>inch</td>
</tr>
<tr>
<td>199</td>
<td>in</td>
</tr>
<tr>
<td>200</td>
<td>feet</td>
</tr>
<tr>
<td>201</td>
<td>ft</td>
</tr>
<tr>
<td>202</td>
<td>yard</td>
</tr>
<tr>
<td>203</td>
<td>mile</td>
</tr>
<tr>
<td>204</td>
<td>psi</td>
</tr>
<tr>
<td>205</td>
<td>ksi</td>
</tr>
<tr>
<td>206</td>
<td>kip</td>
</tr>
<tr>
<td>207</td>
<td>g</td>
</tr>
<tr>
<td>208</td>
<td>pi</td>
</tr>
<tr>
<td>209</td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>NUMBER_OF_NODES</td>
</tr>
<tr>
<td>211</td>
<td>NUMBER_OF_ELEMENTS</td>
</tr>
<tr>
<td>212</td>
<td>CURRENT_TIME</td>
</tr>
<tr>
<td>213</td>
<td>NUMBER_OF_SP_CONSTRAINTS</td>
</tr>
<tr>
<td></td>
<td>NUMBER_OF_MP_CONSTRAINTS</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------</td>
</tr>
<tr>
<td>215</td>
<td>NUMBER_OF_LOADS</td>
</tr>
<tr>
<td>216</td>
<td>IS_PARALLEL</td>
</tr>
<tr>
<td>217</td>
<td>SIMULATE_EXIT_FLAG</td>
</tr>
<tr>
<td>218</td>
<td>then</td>
</tr>
<tr>
<td>219</td>
<td>while</td>
</tr>
<tr>
<td>220</td>
<td>do</td>
</tr>
<tr>
<td>221</td>
<td>let</td>
</tr>
<tr>
<td>222</td>
<td>vector</td>
</tr>
<tr>
<td>223</td>
<td></td>
</tr>
<tr>
<td>224</td>
<td>cos</td>
</tr>
<tr>
<td>225</td>
<td>sin</td>
</tr>
<tr>
<td>226</td>
<td>tan</td>
</tr>
<tr>
<td>227</td>
<td>cosh</td>
</tr>
<tr>
<td>228</td>
<td>sinh</td>
</tr>
<tr>
<td>229</td>
<td>tanh</td>
</tr>
<tr>
<td>230</td>
<td>acos</td>
</tr>
<tr>
<td>231</td>
<td>asin</td>
</tr>
<tr>
<td>232</td>
<td>atan</td>
</tr>
<tr>
<td>233</td>
<td>atan2</td>
</tr>
<tr>
<td>234</td>
<td>sqrt</td>
</tr>
<tr>
<td>235</td>
<td>exp</td>
</tr>
<tr>
<td>236</td>
<td>log10</td>
</tr>
<tr>
<td>237</td>
<td>ceil</td>
</tr>
<tr>
<td>238</td>
<td>fabs</td>
</tr>
<tr>
<td>239</td>
<td>floor</td>
</tr>
<tr>
<td>240</td>
<td>log</td>
</tr>
</tbody>
</table>
205.8 Integrated Development Environment (IDE) for DSL
205.9 Mesh Generation using GiD

1. Download the latest version of GiD from http://www.gidhome.com/, and also get a temporary license (or purchase it...).

2. Download essi.gid.tar.gz, unpack it (tar -xvzf essi.gid.tar.gz) in problemtypes directory that is located in GiD’s root directory.

3. When you run GiD, you will see essi in “Data > Problem types”, and can start using it...

4. A simple movie with instructions for mesh generation is available: (Link to a movie, 11MB).
205.10 Model Development and Mesh Generation using gmesh
205.11 Model Input File Editing using Sublime

http://www.sublimetext.com/
Chapter 206

Output Formats

(In collaboration with Prof. José Abell, Mr. Sumeet Kumar Sinha and Dr. Yuan Feng, and Dr. Han Yang)
206.1 Chapter Summary and Highlights

206.2 Introduction

All output from ESSI simulator is stored inside a database format, specifically designed for handling scientific array-oriented data, called HDF5 (Group, 2020). HDF stands for 'Hierarchical Data Format' and is a self-describing data format suitable for portable sharing of scientific data. The format was created and is maintained by the HDF group (http://www.hdfgroup.org/)

Data is stored within the file using a hierarchy similar to a unix filesystem, with groups to store related data and the actual data stored in so-called ‘datasets’ within each group.

HDF5 was chosen because it meets our design goals of provides:

- A simplified output format. Output is a single HDF5 file per analysis stage.
- An efficient binary (possibly compressed) file format that optimizes random access to data.
- A data format that is amenable to store output from parallel computations.
- Has a reasonable API exposed in several languages so that users can easily and customizably access simulation data.

One very convenient tool for the basic exploration of HDF5 files is the viewer ‘hdfview’ (http://www.hdfgroup.org/products/java/hdfview/index.html).

206.3 Output Filename and Format

On running any simulation on Real-ESSI simulator output files are produced for each analysis stage. The number of outputs and the filename is slightly different for sequential and parallel runs. Each output file, contains the information about the model mesh, nodal displacements, elements output, boundary conditions, material tags.. etc. The output files are designed as completely independent files containing all the data for the loading stage. In parallel each of the follower compute process outputs contains all the data corresponding to only the follower compute process. This is done to make the visualization and output process efficient.

206.3.1 Sequential

For sequential runs a single output file is produced per analysis stage. The files are named according to model and stage names, not by the filename that runs the analysis. The extension is set to be ‘.h5.feioutput’, to distinguish from future possible alternative output formats.
For example, if the model name is ‘site_response’ and the stage name is ‘earthquake_shaking’ the corresponding output filename will be ‘site_response_earthquake_shaking.h5.feioutput’.

206.3.2 Parallel

In parallel, for each stage, output files produced are equal to the number of CPU’s used. For example, a simulation run of 8 CPU’s will produce 8 output files per stage for each corresponding CPU’s (cores). The filename remains the same as sequential output each CPU (process id) used, but the extension is set to be as ‘.h5.pid.feioutput’, where pid refers to the process id of the CPU. However, the main compute process having pid equal to 0 follows the extension ‘.h5.feioutput’. For example, In parallel, if the model name is ‘site_response’ and the stage name is ‘earthquake_shaking’ and the analysis is run on n CPU’s, the corresponding output filename for the main compute process (pid = 0) would be ‘site_response_earthquake_shaking.h5.feioutput’. All the follower compute process having pid > 0 would have output filename ‘site_response_earthquake_shaking.h5.pid.feioutput’.

The main compute process usually does not contain any nodes and elements once the partition is achieved and nodes and elements are transferred to their respective CPU’s or cores. Thus, the output produced by main compute process does not contain any mesh or output results. However, it contains the partition data as shown in Figure 206.1 describing the process id on which any node or element is assigned. This information is quite useful, during post processing when the result of a particular node or element needs to be extracted. The main compute output can be read to find out the follower compute process id on which the data is located and then the output of that process id can be read to get the
data of interest.

## 206.4 Output Units

Real-ESSI .feioutput file stores all the results in standard units. The table below shows the units of all the data stored in HDF5 file.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force $F_x, F_y, F_z$</td>
<td>N</td>
</tr>
<tr>
<td>Moments $M_x, M_y, M_z$</td>
<td>N − m</td>
</tr>
<tr>
<td>Pressure $p$</td>
<td>Pa</td>
</tr>
<tr>
<td>Displacement $u_x, u_y, u_z$</td>
<td>m</td>
</tr>
<tr>
<td>Rotation $r_x, r_y, r_z$</td>
<td>radian</td>
</tr>
<tr>
<td>Stress $\sigma$</td>
<td>Pa</td>
</tr>
<tr>
<td>Strain $\epsilon$</td>
<td>unit less</td>
</tr>
<tr>
<td>Acceleration $a_x, a_y, a_z$</td>
<td>$m/s^2$</td>
</tr>
<tr>
<td>Time $t$</td>
<td>s</td>
</tr>
</tbody>
</table>

## 206.5 Data organization

In HDF5 jargon a multidimensional array is called a **dataset**. Datasets are indexed arrays (up to 32 dimensions) that can contain different types of data. Supported data types are: integers (various sizes), floating point numbers (**float**, **float**, **long double**, etc.), strings of text (fixed and variable size **char** *), and arbitrary structures of data (similar to C language **struct**). A file can contain as many independent datasets as needed. Datasets can be organized into 'groups', which are like folders in a file system. HDF5 provides additionally convenience data-types such as ‘references’, which provide views (slices) into different datasets or portions of them.

In the particular case of the ESSI HDF5 output, the files are designed with the contents and structure explained hereafter and depicted in Figure 206.3.

### 206.5.1 The Root group

The root of the HDF5 file contains information about each stage of loading. In parallel simulations, the information corresponds to the process Id (follower compute node) involved in that stage. The objects under this group are shown in Figure 206.4 and described in List 206.5.1.
Figure 206.2: Output from a typical analysis.

- **time**: (float) A floating point array named which contains the available time steps for this analysis.

- **Number_of_Time_Steps**: (int) A single scalar integer array with the number of times steps.

- **Model_Name**: (string) A single string with the model name.

- **Stage_Name**: (string) A single string with the stage name.

- **PreviousStage**: (string) A single string containing previous stage name.

- **Process_Number**: (int) An integer representing the process id by which output was generated. For sequential runs, process id would be zero. In parallel runs, process id corresponds to the mpi rank or follower compute id of processor involved in computation.

- **Number_of_Processes_Used**: (int) An integer representing total number of processors/CPU/nodes used in the simulation. For sequential runs, it is equal to one whereas for parallel runs, it is
equal to the number of CPU's/Cores used.

- **Number_of_Nodes**: (int) A single scalar integer array with the number of nodes defined in that domain.

- **Number_of_Elements**: (int) A single scalar integer array with the number of elements defined in that domain.
Figure 206.4: Data accessible in the Root directory of HDF5 file.

- **Number_of_Gauss_Points**: (int) A single scalar integer array with the number of gauss points in that domain.

- **Number_of_Element_Outputs**: (int) A single scalar integer array with stores the total length of Element_Output array in that domain.

- **Analysis_Options**: (string) An array of strings with the analysis options selected for the current analysis.

- **Date_and_Time_Start**: (string) A single string with the Date and Time of the start of the analysis. (In Coordinated Universal Time, UTC)

- **Date_and_Time_End**: (string) A single string with the Date and Time of the end of the analysis. (In Coordinated Universal Time, UTC)

- **Version_Info**: (string) A Long string containing the version information of Real-ESSI simulator.

- **Model**: A group that contains the Nodes and Elements groups. It contains essential information about the mesh and analysis results for nodes and elements. See Section 206.5.2
• **Eigen Mode Analysis**: A group that contains the information about the eigen mode analysis results of the domain. See Section 206.5.6

206.5.2 **The Model group**

The Model group contains information about the mesh and analysis outputs. It contains the following groups as shown in Figure 206.5 and is also described below

- **Nodes**: A group that contains the information about the defined nodes and their output for this analysis. See Section 206.5.3
- **Elements**: A group that contains the information about the defined Elements and their output for this analysis. See Section 206.5.4
- **Physical Groups**: A group that contains the information physical group of elements and nodes defined in that domain. See Section 206.5.5
- **Material**: (string) A string array which contains information about the material tag defined in the analysis for that loading stage. Section 206.5.7

![Figure 206.5: Model group directory of HDF5 file.](image)

Subgroups Nodes and Elements store several integer and double precision arrays, that contain all necessary information for post processing.

206.5.3 **The Nodes group**

The Nodes group contains information about the nodal coordinates of the model, their tags, the number of DOFs defined at each node, and the corresponding solution variables (DOF results or generalized displacements) for each time step.

The format used to store the data is designed to give the fastest possible access time to the data of interest. Stored within the Nodes groups (and also in Elements) are two types of arrays: data arrays and index arrays.
• **Data Arrays** :: Data arrays might be floating-point arrays or integer arrays and have names not starting with 'Index_to_'.

• **Index Arrays** :: All index arrays are integer arrays and have names starting with the word 'Index_to_' followed by the name of the array which this array indexes.

The concept of *index array* is an important one regarding speed of access to data. These arrays map the integer tag number of the nodes (or elements) to the data. This allows fast access to components which minimizes searching within arrays to find the data of interest.

The **Nodes** group contains the following index arrays.

• **Index_to_Coordinates**: (int) Indexes the coordinates of nodes. See section 206.5.3.6

• **Index_to_Generalized_Displacements**: (int) Indexes the outputs of nodes (generalized displacements). See section 206.5.3.7

The following are the data arrays available in the **Nodes** group (shown in Figure ?? along with their respective indexing array:

![Figure 206.6: Nodes group directory of HDF5 file.](image)

- **Coordinates**: (float) 1-D array containing nodal coordinates fixed in time. [Indexed by Index_to_Coordinates array]. See section 206.5.3.6

- **Generalized_Displacements**: (float) 2-D array containing the DOF values for the solution at each time step. [Indexed by Index_to_Generalized_Displacements array]. See section 206.5.3.7

- **Number_of_DOFs**: (int) 1-D array mapping the integer tag of each node to the number of DOFs at that node. See section 206.5.3.1
• **Constrained Nodes**: (int) 1-D array mapping the integer tag of each node to the number of DOFs at that node. See section 206.5.3.3

• **Constrained DOFs**: (int) 1-D array mapping the integer tag of each node to the number of DOFs at that node. See section 206.5.3.4

• **Partition**: (int) 1-D array mapping the integer tag of each node to the number of DOFs at that node. See section 206.5.3.2

• **Support_Reactions**: (float) 1-D array mapping the integer tag of each node to the number of DOFs at that node. See section 206.5.3.5

For example, let’s imagine that the user has defined 4 nodes and applied the following constrained as shown below:

```c
// defining nodes
add node # 2 at (0*m, 0*m, 0*m) with 3 dofs;
add node # 4 at (1*m, 1*m, 1*m) with 6 dofs;
add node # 5 at (2*m, 2*m, 2*m) with 3 dofs;
add node # 6 at (1*m, 0*m, 5*m) with 3 dofs;

// applying constraints
fix node # 2 dofs ux uy;
fix node # 4 rx ry rz;
fix node # 6 Ux p;
```

The index and the data arrays for the given example would look like the following as shown in the subsections ahead.

**206.5.3.1 Number_of_DOFs**

*Number_of_DOFs* array defines the number of degrees of freedom for each node defined in the model. It is an integer array of length equal to the maximum node tag + 1 (including tag 0). If a node tag does not exist, the corresponding dofs is output as -1. Figure 206.7 shows how to read *Number_of_DOFs* array. In the given example Listing 206.1, node tag 2 has 3 degrees of freedom. Similarly, node tag 4 has 6 degrees of freedom and so on.

**206.5.3.2 Partition**

*Partition* array contains the domain or process id on which nodes tags were defined in case of a parallel simulation. For sequential runs, this dataset is not available. If a node tag does not exist, the
corresponding partition process id is output as -1. Figure 206.7 shows how to read Partition array. In the given example Listing 206.1, node tag 2 is assigned to process id 1. Similarly, node tag 4 is assigned to process id 2 and so on.

<table>
<thead>
<tr>
<th>Index_no</th>
<th>Number_of_DOFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Node Tag 2 has 3 dofs

<table>
<thead>
<tr>
<th>Index_no</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Node Tag 2 was assigned to Process Id 1

Figure 206.7: Arrays describing node information in Nodes group directory of HDF5 file.

206.5.3.3 Constrained_Nodes

Constrained_Nodes array contains a list of node tags for each dof on which fixities were applied. Figure 206.8 shows how to read Constrained_Nodes. In the given example Listing 206.1, dof $u_x$ and $u_y$ of node tag 2 is fixed. Similarly, for node tag 4 dofs $r_x$ $r_y$ and $r_z$ are fixed. That’s why in the Constrained_Nodes array contains node tags 2,2,4,4,4 and so on multiple times for each dof of the corresponding node tag fixed.

206.5.3.4 Constrained_DOFs

Constrained_DOFs array contains a list of dofs of the corresponding node tag on which fixities were applied. Figure 206.8 shows how to read Constrained_DOFs. In the given example Listing 206.1, dof $u_x$ (0) and $u_y$ (1) of node tag 2 is fixed. Similarly, for node tag 4 dofs $r_x$ (3) $r_y$ (4) and $r_z$ (5) are fixed. Figure 206.8 also show the DOF if numbering for different dof types i.e. for 3dof, 4dof, 6dof and 7dof nodes.

206.5.3.5 Support_Reactions

Support_Reactions contains a (float) array of reaction forces for the constrained degree of freedoms (DOFs). Figure 206.8 shows how to read Constrained_DOFs. In the given example Listing 206.1, dof $u_x$ (0) and $u_y$ (1) of node tag 2 has support reactions 1N and -5N respectively. Similarly, for node tag 4 dofs $r_x$ (3) $r_y$ (4) and $r_z$ (5) have reactions 45, 3 and -5 N – m respectively. The reaction forces for displacement dofs ($u_x,u_y,u_z$) are forces in units of N, for rotational dofs ($r_x,r_y,r_z$) are moments in
### 206.5.3.6 Coordinates

The Coordinates array is a vertical stack of the nodal coordinate values. It is indexed by the Index_to_Coordinates, which relates the integer tag of each node (defined at the moment of creation of every node) with the position on this array of the 3 nodal coordinates. If the node with a given tag is not defined (if a tag number or several are skipped) this array will contain a negative number (-1) for that tag value. Figure 206.9 shows how to read Coordinates and index_to_Coordinates of nodes.

The size of Index_to_Coordinates is always the maximum tag defined plus one (zero can be a tag too). The size of the Coordinates array is three times the number of nodes defined. In the given
Each node has 3 coordinate values ($u_x, u_y$, and $u_z$).

Figure 206.9: Coordinates and Index_to_Coordinates arrays in Nodes group directory of HDF5 file.

e.g. Listing 206.1, the coordinates of node tag 2 is $(0*m, 0*m, 0*m)$. The coordinate of node tag 4 is $(1*m, 1*m, 1*m)$ and so on. The coordinates have unit of meter ($m$).

### 206.5.3.7 Generalized Displacements

The Generalized Displacements array is a 2-D array containing the computed solution at the nodal degrees of freedom for all nodes and all times steps. It is indexed by the Index_to_Generalized_Displacements array (first index) and time (second index). Figure 206.10 shows how to read Generalized Displacements and index_to_Generalized_Displacements of nodes.

Output for displacement dofs ($u_x, u_y, u_z$) are in units of $m$, for rotational dofs ($r_x, r_y, r_z$) are in units of radian and for pressure dof ($p$) in pascal ($Pa$).

With every time step, another column is added to the Generalized Displacements array. This means that the time index (starting at 0) is directly related to the time array at the root of the HDF5 file.

The rows of the Generalized Displacements array contain the results for each DOF of the current node. For a 3-DOF node these will be displacement in $X$, $Y$, and $Z$ ($u_x$, $u_y$, and $u_z$) respectively. For higher number-of-dof nodes the first three components are the same but the remaining ones are going to depend on the connecting elements. For example, 6-DOF nodes might carry information on nodal rotations (beams and shells) or might be fluid displacements in case the elements connecting are $u$–$U$ coupled fluid-porous-solid elements as shown in Figure 206.8. Section 206.6 describes the output definitions for nodes with different dof-types.
Output for Time Step Number 0 also known as 'initial conditions' or 'initial state'.

Index_to_Generalized_Displacements

Index_no (Node_Tag) 0 1 2 3 4 5 6 ...

Generalized_Displacements

Each node has output for their corresponding number of degrees of freedom (DOFs)

Number of rows or outputs per node is given by Number_of_Dofs(Node_Tag)

Output for Time Step Number 0 also known as 'initial conditions' or 'initial state'.

Index_no (Node_Tag) 0 1 2 3 4 5 6 ...

Number_of_Dofs

-1 -1 3 -1 6 3 3 ...

Node tag 2 is mapped to index 0 in Generalized_Displacements array

-1 indicates Node tag 3 does not exist

Node tag 2

Node tag 4

Output for Time Step Number 2

Figure 206.10: The Index_to_Generalized_Displacements and Generalized_Displacements in Nodes group directory of HDF5 file.

206.5.4 The Elements group

The Elements group contains information on the finite element mesh such as: connectivity array, element types, location of Gauss-point integration coordinates (global), materials defined at each element, and all available output from the elements. At this point it is important to note that the kind and amount of output contained in the element output arrays are dependent on element implementation as it is the elements who are in charge of controlling their output. For information on the specific output of each element, consult the documentation for the respective element.

The idea and organization of the datasets contained herein is analogous to the of the Nodes group. This group contains the following index arrays.

- Index_to_Connectivity: (int) Maps element tag to location within the Connectivity array.
- Index_to_Gauss_Point_Coordinates: Maps element tag to location of Gauss-point coordinates in the Gauss_Point_Coordinates array.
- **Index_to_Gauss_Outputs**: (int) Maps element tag to location of gauss output of element in the Gauss_Outputs array.

- **Index_to_Element_Outputs**: (int) Maps element tag to location of output in the Element_Outputs array.

The following are the data arrays available in the Elements group (shown in Figure ?? along with their respective indexing array:

![Elements group directory of HDF5 file.](image)

- **Class_Tags**: (int) It is an array that contains the integer ids for each elements tag defined in the model and present in the domain. See section 206.5.4.4

- **Number_of_NODES**: (int) Maps element tag number to number of nodes in the element (-1 if element tag is not defined). See section 206.5.4.1.

- **Number_of_Gauss_Points**: (int) Maps tag number to the number of Gauss-integration points in that element (-1 if not element tag is not defined). It stores 0 in case of no gauss points (mostly for structural elements). See section 206.5.4.3
• **Material_Tags**: (int) Maps tag number to the tag number of the material contained in that element (-1 if element tag is not defined or material for that element is not defined). See section 206.5.4.6

• **Partition**: (int) Maps tag number to the tag number of the processor id on which it is assigned (-1 if element tag is not defined). See section 206.5.4.5

• **Connectivity**: (int) Contains the nodes tags which are connected by this element [indexed by the Index_to_Connectivity array] (-1 if element tag is not defined). See section 206.5.4.7

• **Gauss_Point_Coordinates**: (float) Contains the coordinates of all the gauss points of that element tag [indexed by the Index_to_Gauss_Point_Coordinates array] (-1 if element tag is not defined). See section 206.5.4.8

• **Gauss_Outputs**: (float) Contains the stress, strain and plastic strain outputs for each gauss point present in the corresponding element tag [indexed by the Index_to_Gauss_Outputs array]. See section 206.5.4.9

• **Element_Outputs**: (float) Contains output other than gauss points by the [indexed by the Index_to_Element_Outputs array]. See section 206.5.4.2

For example, let's imagine that the user has defined 4 elements and some materials as shown below:

```
Listing 206.2: Element_Example

// defining materials
add material #1 type uniaxial_elastic elastic_modulus = 1*Pa ← viscoelastic_modulus = 0*Pa*s;
add material #2 type linear_elastic_isotropic_3d mass_density = 2000*kg/m^3 ← elastic_modulus = 200*MPa poisson_ratio = 0.3;

// defining elements
add element #2 type truss with nodes (1,2) use material # 1 cross_section = ← 1*m^2 mass_density = 0*kg/m^3;
add element #4 type 8NodeBrick with nodes (1,8,6, 4, 3, 9, 2, 5) use material #2;
add element #5 type HardContact with nodes (3,2) normal_stiffness = 1e10*N/m ← tangential_stiffness = 1e4*Pa*m normal_damping = 0*kN/m*s ← tangential_damping = 0*N/m*s friction_ratio = 1 contact_plane_vector = ← (0,0,1);
add element #6 type 8NodeBrick with nodes (11,18,61,14, 3,19,22,15) use ← material #2;
```

The index and the data arrays for the given example would look like the following as shown in the subsections ahead.
206.5.4.1 Number_of_Nodes

Number_of_Nodes array defines the number of nodes for each element defined in the model. It is an integer array of length equal to the maximum element tag + 1 (including tag 0). If a element tag does not exists, the corresponding number of nodes is output as -1. Figure 206.12 shows how to read Number_of_Nodes array. In the given example Listing 206.2, element tag 2 has 2 nodes. Similarly, element tag 4 has 8 nodes and so on.

206.5.4.2 Number_of_Element_Outputs

Number_of_Element_Outputs array defines the number of outputs for each element defined in the model. It is an integer array of length equal to the maximum element tag + 1 (including tag 0). If a element tag does not exists, the corresponding number of outputs is stored as -1. Figure 206.12 shows how to read Number_of_Element_Outputs array. In the given example Listing 206.2, element tag 2 has 2 outputs. Similarly, element tag 5 has 9 outputs but element tag 4 has 0 outputs and so on.

Figure 206.12: Arrays describing element information in Elements group directory of HDF5 file.
206.5.4.3 Number_of_Gauss_Points

Number_of_Gauss_Points array defines the number of gauss points for each element defined in the model. It is an integer array of length equal to the maximum element tag + 1 (including tag 0). If a element tag does not exists, the corresponding number of outputs is stored as -1. Figure 206.12 shows how to read Number_of_Gauss_Points array. In the given example Listing 206.2, element tag 2 and 5 has 0 gauss points. Similarly, element tag 4 has 8 gauss points and so on.

206.5.4.4 Class_Tags

Class_Tags array defines an (unique) element type for each of the element defined in the model. It is an integer array of length equal to the maximum element tag + 1 (including tag 0). If a element tag does not exists, the corresponding number of outputs is stored as -1. Figure 206.12 shows how to read Class_Tags array. In the given example Listing 206.2, element tag 2 has class tag of 88 (i.e. truss element). Similarly, element tag 5 has class tag 2 (i.e. 8 node brick element) and so on. Table 206.1 and Table 206.2 shows class tags for different element types.

206.5.4.5 Partition

Partition array contains the domain or process id on which element tags were defined in case of a parallel simulation. For sequential runs, this dataset is not available. If a element tag does not exists, the corresponding partition process id is output as -1. Figure 206.12 shows how to read Partition array. In the given example Listing 206.2, element tag 2 is assigned to process id 2. Similarly, element tag 4 is assigned to process id 1 and so on.

Figure 206.13: Arrays describing node information in Nodes group directory of HDF5 file.
206.5.4.6 Material_Tags

Material_Tags array defines the material tag number for each element defined in the model. It is an integer array of length equal to the maximum element tag + 1 (including tag 0). If a element tag does not exists, the corresponding number of outputs is stored as -1. Figure 206.12 shows how to read Material_Tags array. In the given example Listing 206.2, element tag 2 has material tag of 1. Similarly, element tag 4 has material tag 2. Whereas, element tag 5 have material tag of -1.
### Table 206.2: Class Tags for Real-ESSI Elements (Part 2)

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Class Tag</th>
<th>Element Type</th>
<th>Class Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>EightNodeBrickOrderFour</td>
<td>50</td>
<td>EightNodeBrickOrderFour_up</td>
<td>51</td>
</tr>
<tr>
<td>EightNodeBrickOrderFour_upU</td>
<td>52</td>
<td>TwentyNodeBrickOrderFour</td>
<td>53</td>
</tr>
<tr>
<td>TwentyNodeBrickOrderFour_up</td>
<td>54</td>
<td>TwentyNodeBrickOrderFour_upU</td>
<td>55</td>
</tr>
<tr>
<td>TwentySevenNodeBrickOrderFour</td>
<td>56</td>
<td>TwentySevenNodeBrickOrderFour_up</td>
<td>57</td>
</tr>
<tr>
<td>TwentySevenNodeBrickOrderFour_upU</td>
<td>58</td>
<td>VariableNodeBrickOrderFour</td>
<td>59</td>
</tr>
<tr>
<td>VariableNodeBrickOrderFour_up</td>
<td>60</td>
<td>VariableNodeBrickOrderFour_up</td>
<td>61</td>
</tr>
<tr>
<td>EightNodeBrickOrderFive</td>
<td>62</td>
<td>EightNodeBrickOrderFive_up</td>
<td>63</td>
</tr>
<tr>
<td>EightNodeBrickOrderFive_upU</td>
<td>64</td>
<td>TwentyNodeBrickOrderFive</td>
<td>65</td>
</tr>
<tr>
<td>TwentyNodeBrickOrderFive_up</td>
<td>66</td>
<td>TwentyNodeBrickOrderFive_upU</td>
<td>67</td>
</tr>
<tr>
<td>TwentySevenNodeBrickOrderFive</td>
<td>68</td>
<td>TwentySevenNodeBrickOrderFive_up</td>
<td>69</td>
</tr>
<tr>
<td>TwentySevenNodeBrickOrderFive_upU</td>
<td>70</td>
<td>VariableNodeBrickOrderFive</td>
<td>71</td>
</tr>
<tr>
<td>VariableNodeBrickOrderFive_up</td>
<td>72</td>
<td>VariableNodeBrickOrderFive_upU</td>
<td>73</td>
</tr>
<tr>
<td>EightNodeBrickOrderSix</td>
<td>74</td>
<td>EightNodeBrickOrderSix_up</td>
<td>75</td>
</tr>
<tr>
<td>EightNodeBrickOrderSix_upU</td>
<td>76</td>
<td>TwentyNodeBrickOrderSix</td>
<td>77</td>
</tr>
<tr>
<td>TwentyNodeBrickOrderSix_up</td>
<td>78</td>
<td>TwentyNodeBrickOrderSix_upU</td>
<td>79</td>
</tr>
<tr>
<td>TwentySevenNodeBrickOrderSix</td>
<td>80</td>
<td>TwentySevenNodeBrickOrderSix_up</td>
<td>81</td>
</tr>
<tr>
<td>TwentySevenNodeBrickOrderSix_upU</td>
<td>82</td>
<td>VariableNodeBrickOrderSix_up</td>
<td>83</td>
</tr>
<tr>
<td>VariableNodeBrickOrderSix_up</td>
<td>84</td>
<td>VariableNodeBrickOrderSix_upU</td>
<td>85</td>
</tr>
<tr>
<td>HardContact</td>
<td>86</td>
<td>SoftContact</td>
<td>87</td>
</tr>
<tr>
<td>Truss</td>
<td>88</td>
<td>ElasticBeam</td>
<td>89</td>
</tr>
<tr>
<td>ThreeNodeAndesShell</td>
<td>90</td>
<td>FourNodeAndesShell</td>
<td>91</td>
</tr>
<tr>
<td>ShearBeam</td>
<td>92</td>
<td>rank_one_deficient_elastic_pinned_fixed_beam</td>
<td>93</td>
</tr>
<tr>
<td>DispBeamColumn3d</td>
<td>94</td>
<td>Cosserat_8node_brick</td>
<td>95</td>
</tr>
</tbody>
</table>

### 206.5.4.7 Connectivity

Connectivity array stores the list of node tags in the same order as they were defined along with element declaration in the model. It is an integer. It is indexed by the Index_to_Connectivity array and the number of rows is defined by Number_of_Nodes. Figure 206.14 shows how to read Connectivity array for a particular element. In the given example Listing 206.2, element tag 2 includes node tag 1 and 2. Similarly, element tag 4 includes 8 node tags 1, 8, 6, 4, 3, 9, 2 and 5 respectively.
Figure 206.14: Arrays describing connectivity information in Elements group directory of HDF5 file.

### 206.5.4.8 Gauss_Point_Coordinates

Gauss_Point_Coordinates array stores the coordinates of the gauss points declared inside for each element defined in the model. It is a float array indexed by the Index_to_Gauss_Point_Coordinates array and the number of rows is defined by $3 \times \text{Number of Gauss Points}$. Figure 206.15 shows how to read Gauss_Point_Coordinates array for a particular element. In the given example Listing 206.2, element tag 2 and 5 has no gauss points. Since, element tag 4 has 8 gauss points, the total length of gauss point coordinates output for that element is $8 \times 3 = 24$. The index from which the coordinates information start is 0. Coordinate values for first 3 index $(0, 1, 2)$ corresponds to gauss point 1 and next 3 index $(3, 4, 5)$ corresponds to gauss point 2 and so on.

### 206.5.4.9 Gauss_Output

Gauss_Output array stores the coordinates of the gauss points declared inside for each element defined in the model. It is a 2D float array indexed by the Index_to_Gauss_Outputs array and the number of rows is defined by $18 \times \text{Number of Gauss Points}$. The column index is represented by the time step of the simulation. Time index 0 represents initial state conditions, i.e. the state before the start of new stage and end of previous stage.

Figure 206.16 shows how to read Gauss_Output array for a particular element. In the given example Listing 206.2, element tag 2 and 5 has no gauss points. Since, element tag 4 has 8 gauss points, the...
Figure 206.15: Arrays describing gauss coordinates information in Elements group directory of HDF5 file.

Figure 206.16: Arrays describing gauss output information in Elements group directory of HDF5 file.
The total length of gauss output for that element is $8 \times 18 = 144$. The index from which the coordinates information start is 0. Gauss values for first 18 index ($0, 1..17$) corresponds to gauss point 1 and next 18 index ($18..35$) corresponds to gauss point 2 and so on. For gauss point 1, first 6 index ($0, 1..5$) corresponds to total strain, next 6 index ($6..11$) corresponds to plastic strain and next 6 index ($12..17$) corresponds to stress. Section 206.7 describes how gauss output is stored and what are the units and meaning of those outputs.

### 206.5.4.10 Element Outputs

Element Outputs array stores output for the elements except those stored at gauss points (i.e. stress, total strain and plastic strain). It is a 2D float array indexed by the Index_to_Element_Outputs array and the number of rows is defined Number_of_Element_Outputs. The column index is represented by the time step of the simulation. Time index 0 represents initial state conditions, i.e. the state before the start of new stage and end of previous stage.

Figure 206.17: Arrays describing output information in Elements group directory of HDF5 file.

Figure 206.17 shows how to read Element Outputs array for a particular element and particular
time step. In the given example Listing 206.2, element tag 2 has two outputs. Similarly, element tag 5 has 9 outputs. On the other hand element tag 4 has no output. For element 2, the output starts at row index 0 and ends at 1. Similarly, for element tag 4, the output is stored in row index 2..10.

Since the Element Outputs array format depends on the elements present in the model, one must refer to each element specifically (Section 206.8) to identify what each output component means.

206.5.5 The Physical Groups group

Physical Groups group contains the physical groups of nodes or elements defined in the analysis. It contains two subgroups: Physical Node Groups and Physical Element Groups as shown in Figure 206.18 and is described in the following sections.

Figure 206.18: Physical Groups group directory of HDF5 file.

For example, lets imagine that the user has defined one physical group of nodes and elements respectively as shown below:

Listing 206.3: PhyGrp_Example

```plaintext
// defining physical groups
define physical_node_group "Physical_Node_Group#1";
define physical_element_group "Physical_Element_Group#1";

// adding items to already defined physical groups
add nodes (1,4,5,7,2,30,42) to physical_node_group "Physical_Node_Group#1"
add elements (1,4,5,7,2,30,42) to physical_node_group "Physical_Element_Group#1"
```

The data arrays for the given example would look like the following as shown in the subsections ahead.

206.5.5.1 The Physical Element Groups

This group contains information about physical groups of elements defined in the analysis. For each of the physical group defined, a new integer data array is created inside Physical Element Groups which stored the element tags belonging to that group. Figure 206.19 shows how to read Physical Element Groups dataset array for a particular defined physical group.
In the given example Listing 206.3, a physical group array with name ‘‘Physical_Element_Group#1’’ is created which contains the list of element tags that were part of that physical group. Figure 206.19 shows that ‘‘Physical_Element_Group#1’’ contains element tags 1,4,5,7, and so on.

Figure 206.19: Arrays describing physical groups information in Physical_Groups group directory of HDF5 file.

206.5.5.2 The Physical_Node_Groups

This group contains information about physical groups of nodes defined in the analysis. For each of the physical group defined, a new integer data array is created inside Physical_Node_Groups which stored the node tags belonging to that group. Figure 206.19 shows how to read Physical_Node_Groups dataset array for a particular defined physical group.

In the given example Listing 206.3, a physical group array with name ‘‘Physical_Node_Group#1’’ is created which contains the list of node tags that were part of that physical group. Figure 206.19 shows that ‘‘Physical_Node_Group#1’’ contains node tags 1,4,5,7, and so on;

206.5.6 The Eigen_Mode_Analysis group

Eigen_Mode_Analysis group gets created in HDF5 .feioutput file after an eigen mode analysis. The data arrays available inside this group are described below and is also shown in Figure 206.20.

206.5.6.1 Number_of_Eigen_Modes

Number_of_Modes is an integer that stores information about the number of modes solved for the eigen problem. Figure 206.21 shows how to read it.
206.5.6.2 Eigen_Frequencies

Frequencies is an float array that stores the natural frequencies of the model corresponding to different modes. The length of the array is number of modes + 1. Index ‘0’ does not correspond to any eigen mode and thus stores -1. Figure 206.21 shows how to read it.

```
Index (Eigen_Mode_No) 0 1 2 3 4 5 6 ...
Eigen_Frequencies 0 0.5 1 1 2 2 10 ...
```

Frequency for mode 2

```
Index (Eigen_Mode_No) 0 1 2 3 4 5 6 ...
Eigen_Periods 0 2 1 1 0.5 0.5 0.1 ...
```

Period for mode 2

```
Index (Eigen_Mode_No) 0 1 2 3 4 5 6 ...
Eigen_Values 0 4 5 7 2 30 42 ...
```

Eigen value for mode 2

Figure 206.21:  Eigen_Mode_Analysis group directory of HDF5 file.

206.5.6.3 Eigen_Periods

Frequencies is an float array that stores the natural periods of the model corresponding to different modes. The length of the array is number of modes + 1. Index ‘0’ does not correspond to any eigen mode and thus stores -1. Figure 206.21 shows how to read it.

206.5.6.4 Eigen_Values

Frequencies is an float array that stores the natural eigen values of the model corresponding to different modes. The length of the array is number of modes + 1. Index ‘0’ does not correspond to any
eigen mode and thus stores -1. Figure 206.21 shows how to read it.

206.5.6.5 Modes

Modes is a 2-D float array that stores the generalized displacements of the nodes defined in the model corresponding to different modes. The column index no. n this 2-D array defines the mode no i.e. column index 1 corresponds to mode number 1 and so on. Figure 206.21 shows how to read it.

![Diagram of Modes array](image)

Figure 206.22: Eigen Mode Analysis group directory of HDF5 file.

206.5.7 The Material data array

Material is a 1-D string array that stores information about all materials defined in the model. The length of the array is the maximum material tag + 1 (including tag 0). The index of this array corresponds to the material tag. A value of '-1' means that the material tag was not defined. Figure 206.23 shows the Material data array for the example Listing 206.2. Here, index no 1 and index no 2 stores the information corresponding to material tag 1 and 2 respectively.
206.6 Node-specific output format

In Real-ESSI simulator, nodes can be defined with different number of degree of freedoms (DOFs). Nodes with different dofs can be thought of different types of nodes. As a result, their corresponding output also changes. Here is described, all the different node types available and the output format and their descriptions for each of them. The following subsections would describe the definition of outputs that are expected in Generalized_Displacements and Support_Reactions data arrays in Nodes group of HDF5 output file for node tags of different dof types.

206.6.1 3DOF

Nodes defined with 3DOF type has $u_x, u_y, u_z$ degrees of freedom. They correspond to the displacement degrees of freedom in x, y and z direction respectively. The dof id, generalized.displacement and support reactions for 3dof type node are summarized in the Table 206.6.1.

<table>
<thead>
<tr>
<th>DOF Id</th>
<th>Description</th>
<th>Generalized_Displacements</th>
<th>Support_Reactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>disp. in x-dir</td>
<td>$u_x[m]$</td>
<td>$F_x[N]$</td>
</tr>
<tr>
<td>1</td>
<td>disp. in y-dir</td>
<td>$u_y[m]$</td>
<td>$F_y[N]$</td>
</tr>
<tr>
<td>2</td>
<td>disp. in z-dir</td>
<td>$u_z[m]$</td>
<td>$F_z[N]$</td>
</tr>
</tbody>
</table>

Table 206.3: DOF id and output for 3DOF type node

206.6.2 4DOF

Nodes defined with 3DOF type has $u_x, u_y, u_z, p$ degrees of freedom. They correspond to the displacement degrees of freedom in x, y and z direction and pressure respectively. The dof id, generalized.displacement and support reactions for 4dof type node are summarized in the Table 206.6.2. $up$ elements have nodes
nodes defined with 6DOF type has \( u_x, u_y, u_z, r_x, r_y, r_z \) degrees of freedom. They correspond to the displacement and rotational degrees of freedom in \( x, y \) and \( z \) direction respectively. The dof id, generalized_displacement and support reactions for 6 dof type node are summarized in the Table 206.6.3. Beam and Shell elements have nodes with 6 dofs.

<table>
<thead>
<tr>
<th>DOF Id</th>
<th>Description</th>
<th>Generalized_Displacements</th>
<th>Support_Reactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>disp. in x-dir</td>
<td>( u_x[m] )</td>
<td>( F_x[N] )</td>
</tr>
<tr>
<td>1</td>
<td>disp. in y-dir</td>
<td>( u_y[m] )</td>
<td>( F_y[N] )</td>
</tr>
<tr>
<td>2</td>
<td>disp. in z-dir</td>
<td>( u_z[m] )</td>
<td>( F_z[N] )</td>
</tr>
<tr>
<td>3</td>
<td>rotation about x-axis</td>
<td>( r_x[\text{radian}] )</td>
<td>( M_x[N-m] )</td>
</tr>
<tr>
<td>4</td>
<td>rotation about y-axis</td>
<td>( r_y[\text{radian}] )</td>
<td>( M_y[N-m] )</td>
</tr>
<tr>
<td>5</td>
<td>rotation about z-axis</td>
<td>( r_z[\text{radian}] )</td>
<td>( M_z[N-m] )</td>
</tr>
</tbody>
</table>

Table 206.5: DOF id and output for 6DOF type node

206.6.4  7DOF

Nodes defined with 7DOF type has \( u_x, u_y, u_z, p, U_x, U_y, U_z \) degrees of freedom. They correspond to the solid-displacement, pore-fluid pressure and fluid-displacement in \( x, y \) and \( z \) direction respectively. The dof id, generalized_displacement and support reactions for 7dof type node are summarized in the Table 206.6.3. upU elements have nodes with 7 dofs.
206.7 Element-gauss output format

Gauss_Outputs array stores the coordinates of the gauss points declared inside for each element defined in the model. It is a 2D float array indexed by the Index_to_Gauss_Outputs array and the number of rows is defined by 18 (6 total strain, 6 plastic strain and 6 stress) times Number_of_Gauss_Points. The column index is represented by the time step of the simulation. Time index 0 represents initial state conditions, i.e. the state before the start of new stage and end of previous stage.

Figure 206.24 shows how to read Gauss_Outputs array for a particular element. In the given example Listing 206.2, element tag 2 and 5 has no gauss points. Since, element tag 4 has 8 gauss points, the total length of gauss output for that element is $8 \times 18 = 144$. The index from which the coordinates information start is 0. gauss values for first 18 index (0,1..17) corresponds to gauss point 1 and next 18 index (18..35) corresponds to gauss point 2 and so on. For gauss point 1, first 6 index (0,1..5) corresponds to total strain, next 6 index (6..11) corresponds to plastic strain and next 6 index (12..17) corresponds to stress. Table 206.7 shows how data is stored for each gauss points. The strains are unit less and stress have unit of Pascal Pa.

In the Table 206.7 start corresponds to the starting position for the elements output as determined with reference to the Index_to_Element_Outputs for the element of interest. To this number we add the corresponding offset as determined by each table below and interpret the row according to the meaning established below.

For each gauss outputs there are 6 components of strain tensor, 6 components of plastic-strain tensor and 6 components of stress tensor. This makes a total of $3 \times 6 \times \text{NumGauss}$ rows of output per element. The specific meaning of the rows is as follows. For Gauss-point 1 (with starting index given in the Index_to_Gauss_Outputs array), the outputs are stored as the following:

<table>
<thead>
<tr>
<th>DOF Id</th>
<th>Description</th>
<th>Generalized_Displacements</th>
<th>Support_Reactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>solid disp. in x-dir</td>
<td>$u_x[m]$</td>
<td>$F_s^x[N]$</td>
</tr>
<tr>
<td>1</td>
<td>solid disp. in y-dir</td>
<td>$u_y[m]$</td>
<td>$F_s^y[N]$</td>
</tr>
<tr>
<td>2</td>
<td>solid disp. in z-dir</td>
<td>$u_z[m]$</td>
<td>$F_s^z[N]$</td>
</tr>
<tr>
<td>3</td>
<td>pore-pressure</td>
<td>$p[Pa]$</td>
<td>$p[Pa]$</td>
</tr>
<tr>
<td>4</td>
<td>fluid disp. in x-dir</td>
<td>$U_x[m]$</td>
<td>$F_f^x[N]$</td>
</tr>
<tr>
<td>5</td>
<td>fluid disp. in y-dir</td>
<td>$U_y[m]$</td>
<td>$F_f^y[N]$</td>
</tr>
<tr>
<td>6</td>
<td>fluid disp. in z-dir</td>
<td>$U_z[m]$</td>
<td>$F_f^z[N]$</td>
</tr>
</tbody>
</table>

Table 206.6: DOF id and output for 7DOF type node
Each gauss point has 18 output 6 total strain, 6 plastic strain and 6 stress
Number of rows or outputs per node is given by Number_of_Gauss_points(Element_Tag) x 18

Output for Time Step Number 0 also known as ‘initial conditions’ or ‘initial state’

Number of Gauss Points is 8 for element tag 4, which means total number of outputs would be 18x8=144

Table 206.7: Output Format for each gauss point defined inside element
Gauss-point 2 will then start at position 18 through 36 with the same meaning for each row. And so-on for the other Gauss-points.
206.8 Element-specific output format

Element_Outputs array stores output for the elements except those stored at gauss points (i.e. stress, total strain and plastic strain). It is a 2D float array indexed by the Index_to_Element_Outputs array and the number of rows is defined by Number_of_Element_Outputs. The column index is represented by the time step of the simulation. Time index 0 represents initial state conditions, i.e. the state before the start of new stage and end of previous stage.

![Diagram of Element_Outputs array and Index_to_Element_Outputs array]

-1 indicates Element tag 3 does not exist.

Figure 206.25: Arrays describing output information in Elements group directory of HDF5 file.

Figure 206.25 shows how to read Element_Outputs array for a particular element and particular time step. In the given example Listing 206.2, element tag 2 has two outputs. Similarly, element tag 5 has 9 outputs. On the other hand element tag 4 has no output. For element 2, the output starts at row index 0 and ends at 1. Similarly, for element tag 4, the output is stored in row index 2..10.

Since the Element_Outputs array format depends on the elements present in the model, one must...
refer to each element specifically. Each element writes information into the Element Outputs array in a different way. The user can determine the rows of the Element Outputs array that belong to a given element by looking into the Index to Element Outputs array which relates element tag to the starting position of that element’s output within the Element Outputs array. Additionally, the Number of Element Outputs tells the user how many rows after the starting position correspond to the given element output.

The actual meaning of each row is element dependent and is detailed in the following pages. Please note that elements not in this list have no output defined at the moment.

In what follows start corresponds to the starting position for the elements output as determined with reference to the Index to Element Outputs for the element of interest. To this number we add the corresponding offset as determined by each table below and interpret the row according to the meaning established below.

### 206.8.1 Truss

This element outputs 2 rows in total. First row is the uniaxial change in length while second component is the axial force. Truss does not have any gauss points and thus do not have any gauss outputs. The description of each of these rows are shown in Table 206.8:

<table>
<thead>
<tr>
<th>Position (start+offset)</th>
<th>Content</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>start+0</td>
<td>ΔL[m]</td>
<td></td>
</tr>
<tr>
<td>start+1</td>
<td>ForceF[N]</td>
<td></td>
</tr>
</tbody>
</table>

Table 206.8: Element Output description for Truss

### 206.8.2 Brick Elements

All the bricks (u, up, upU and variable) have 0 element outputs. But, they do output total strain, plastic strain and stress for all their corresponding number of gauss points involved in elasto-plastic integration. The format in which gauss outputs are stored for each gauss point is described in Section 206.8

### 206.8.3 ShearBeam

Like brick elements, Shear Beam element also does not have any element output. But it does output for the one gauss point it has. Thus, in total it has 18 outputs for the only one gauss point. The format in
which gauss outputs are stored for each gauss point is described in Section 206.8

206.8.4 ElasticBeam

This element outputs 6 components of local nodal displacements at each of its two nodes, and 6 components of end forces at each of its two nodes. Total number of outputs is thus $2 \times 6 \times 2 = 24$ rows per element. The description of each of these rows are shown in Table 206.9:

<table>
<thead>
<tr>
<th>Position</th>
<th>Content</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>start+0</td>
<td>$u_{x,1}^\text{local}$</td>
<td>$[m]$</td>
</tr>
<tr>
<td>start+1</td>
<td>$u_{y,1}^\text{local}$</td>
<td>$[m]$</td>
</tr>
<tr>
<td>start+2</td>
<td>$u_{z,1}^\text{local}$</td>
<td>$[m]$</td>
</tr>
<tr>
<td>start+3</td>
<td>$\theta_{x,1}^\text{local}$</td>
<td>$[\text{radian}]$</td>
</tr>
<tr>
<td>start+4</td>
<td>$\theta_{y,1}^\text{local}$</td>
<td>$[\text{radian}]$</td>
</tr>
<tr>
<td>start+5</td>
<td>$\theta_{z,1}^\text{local}$</td>
<td>$[\text{radian}]$</td>
</tr>
<tr>
<td>start+6</td>
<td>$u_{x,2}^\text{local}$</td>
<td>$[m]$</td>
</tr>
<tr>
<td>start+7</td>
<td>$u_{y,2}^\text{local}$</td>
<td>$[m]$</td>
</tr>
<tr>
<td>start+8</td>
<td>$u_{z,2}^\text{local}$</td>
<td>$[m]$</td>
</tr>
<tr>
<td>start+9</td>
<td>$\theta_{x,2}^\text{local}$</td>
<td>$[\text{radian}]$</td>
</tr>
<tr>
<td>start+10</td>
<td>$\theta_{y,2}^\text{local}$</td>
<td>$[\text{radian}]$</td>
</tr>
<tr>
<td>start+11</td>
<td>$\theta_{z,2}^\text{local}$</td>
<td>$[\text{radian}]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Position</th>
<th>Content</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>start+12</td>
<td>$F_{x,1}^\text{local}$</td>
<td>$[N]$</td>
</tr>
<tr>
<td>start+13</td>
<td>$F_{y,1}^\text{local}$</td>
<td>$[N]$</td>
</tr>
<tr>
<td>start+14</td>
<td>$F_{z,1}^\text{local}$</td>
<td>$[N]$</td>
</tr>
<tr>
<td>start+15</td>
<td>$M_{x,1}^\text{local}$</td>
<td>$[N \cdot m]$</td>
</tr>
<tr>
<td>start+16</td>
<td>$M_{y,1}^\text{local}$</td>
<td>$[N \cdot m]$</td>
</tr>
<tr>
<td>start+17</td>
<td>$M_{z,1}^\text{local}$</td>
<td>$[N \cdot m]$</td>
</tr>
<tr>
<td>start+18</td>
<td>$F_{x,2}^\text{local}$</td>
<td>$[N]$</td>
</tr>
<tr>
<td>start+19</td>
<td>$F_{y,2}^\text{local}$</td>
<td>$[N]$</td>
</tr>
<tr>
<td>start+20</td>
<td>$F_{z,2}^\text{local}$</td>
<td>$[N]$</td>
</tr>
<tr>
<td>start+21</td>
<td>$M_{x,2}^\text{local}$</td>
<td>$[N \cdot m]$</td>
</tr>
<tr>
<td>start+22</td>
<td>$M_{y,2}^\text{local}$</td>
<td>$[N \cdot m]$</td>
</tr>
<tr>
<td>start+23</td>
<td>$M_{z,2}^\text{local}$</td>
<td>$[N \cdot m]$</td>
</tr>
</tbody>
</table>

Table 206.9: Element Output description for Elastic Beam

206.8.5 4NodeShell_ANDES

This element currently does not have any gauss or element outputs.

206.8.6 BeamColumnDispFiber3d

This element outputs 6 components of end forces at each of its two nodes. Each row is described in Table 206.10:
206.8.7 Contact/Interface Elements

This element outputs 9 components: 3 components of gap displacement and 3 components of contact/interface forces and 3 components of incremental slip in the local axis definition. The first two components of gap are transverse components \( g_1, g_2 \), while the third is the normal gap component \( g_n \). Similarly, the first two components of force \( F_1, F_2 \) are transverse (shear on contact/interface plane) while the third is the normal contact/interface force \( F_n \). The last three components are incremental slip \( \Delta g_{inc,1}, \Delta g_{inc,2} \) in the local transverse direction and total uplift \( \Delta n \) in local normal vector direction.

If \( \Delta n > 0 \), there is uplift i.e. loss of contact/interface else it is in contact. Each row is described in Table 206.11:

<table>
<thead>
<tr>
<th>Position (start+offset)</th>
<th>Content meaning</th>
<th>Position (start+offset)</th>
<th>Content meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>start+0</td>
<td>( F_{x,1}^{local}[N] )</td>
<td>start+6</td>
<td>( F_{x,2}^{local}[N] )</td>
</tr>
<tr>
<td>start+1</td>
<td>( F_{y,1}^{local}[N] )</td>
<td>start+7</td>
<td>( F_{y,2}^{local}[N] )</td>
</tr>
<tr>
<td>start+2</td>
<td>( F_{z,1}^{local}[N] )</td>
<td>start+8</td>
<td>( F_{z,2}^{local}[N] )</td>
</tr>
<tr>
<td>start+3</td>
<td>( M_{x,1}^{local}[N-m] )</td>
<td>start+9</td>
<td>( M_{x,2}^{local}[N-m] )</td>
</tr>
<tr>
<td>start+4</td>
<td>( M_{y,1}^{local}[N-m] )</td>
<td>start+10</td>
<td>( M_{y,2}^{local}[N-m] )</td>
</tr>
<tr>
<td>start+5</td>
<td>( M_{z,1}^{local}[N-m] )</td>
<td>start+11</td>
<td>( M_{z,2}^{local}[N-m] )</td>
</tr>
</tbody>
</table>

Table 206.10: Element Output description for displacement based fiber beams

<table>
<thead>
<tr>
<th>Position (start+offset)</th>
<th>Content meaning</th>
<th>Position (start+offset)</th>
<th>Content meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>start+0</td>
<td>( g_1[m] )</td>
<td>start+3</td>
<td>( F_{11}[N] )</td>
</tr>
<tr>
<td>start+1</td>
<td>( g_2[m] )</td>
<td>start+4</td>
<td>( F_{12}[N] )</td>
</tr>
<tr>
<td>start+2</td>
<td>( g_n[m] )</td>
<td>start+5</td>
<td>( F_n[N] )</td>
</tr>
<tr>
<td>start+6</td>
<td>( \Delta g_{inc,1}[m] )</td>
<td>start+7</td>
<td>( \Delta g_{inc,2}[m] )</td>
</tr>
<tr>
<td>start+8</td>
<td>( Uplift \Delta g_n[m] )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 206.11: Element Output description for contact/interface elements
206.9 Energy output format

Energy folder in the output file stores simulation results for energy-related quantities. Energy density quantities are calculated at element integration points. Averaged energy density quantities are calculated for each element. Energy quantities are calculated for each element. Nodal input energy values are calculated at each node. All energy terms are calculated and stored at each time step.

Under the Energy folder, there are currently seven datasets. Four of them contain energy calculation results. The other three are index datasets, which can be used to find energy output for specific elements or nodes.

In general, energy output datasets are 2D float arrays. The column index is represented by the time step of the simulation. The row index represents various energy quantities that are calculated and stored. The three index datasets are used to locate the row index for specific energy quantities.

206.9.1 Input Energy

All types of nodal load, including DRM, are applied at nodes. Elemental loads are automatically converted to nodal loads, then also applied at nodes. Therefore, input energy or input work from all types of external load can be calculated at each node.

Under the Energy folder, the Nodal_Input_Energy dataset stores input energy at each node. Note that this is input energy accumulated from the start of simulation to the current time step. To find the input energy time history at a specific node, find the index in the Index_to_Nodal_Input_Energy dataset then find the corresponding entry in the Nodal_Input_Energy dataset.

For example, here are the steps to get the input energy time history at the node with node tag 73, Go to Index_to_Nodal_Input_Energy, find the index for node 73 to be 105. Then the accumulated input energy at your chosen node is stored at row 105 in Nodal_Input_Energy.

206.9.2 Energy Density Quantity at Gauss Point

Energy_Density_GP dataset stores energy density quantities at each Gauss point. To find the energy density time history at a specific Gauss point, find the index in the Index_to_Energy_Density_GP dataset then find the corresponding entry in the Energy_Density_GP dataset.

For each Gauss point, 12 data slots are occupied in the following order:

- Incremental kinetic energy density
- Accumulated kinetic energy density
- Incremental strain energy density
- Accumulated strain energy density
- Incremental plastic free energy density
- Accumulated plastic free energy density
- Incremental plastic dissipation density
- Accumulated plastic dissipation density
- The last 4 slots are currently empty, reserved for potential future use

Note that "incremental" means the change of energy density during the time step, "accumulated" means current cumulative energy density at the time step.

Here is an example showing how to obtain the energy density evolution at a specific Gauss point. Say you are interested in looking at the accumulated plastic dissipation density at the third Gauss point of an 8NodeBrick element with element tag 73. Go to Index_to_Energy_Density_GP, find the index for element 73 to be 3904. Since each Gauss point occupies 12 slots in Energy_Density_GP, the energy outputs for the third Gauss point starts at index $3904 + (3-1) \times 12 = 3928$. Finally, since accumulated plastic dissipation density is the 8th entry among the 12 slots, the row index for the data of your interest is $3928 + (8-1) = 3935$. This means that the accumulated plastic dissipation density at your chosen location is stored at row 3935 of Energy_Density_GP.

### 206.9.3 Average Energy Density Quantity of Element

Energy_Density_Element_Average dataset stores averaged energy density quantities of each element. To find the energy density time history of a specific element, find the index in the Index_to_Energy_Element dataset then find the corresponding entry in the Energy_Density_Element_Average dataset.

For each element, 12 data slots are occupied in the following order:

- Incremental kinetic energy density
- Accumulated kinetic energy density
- Incremental strain energy density
- Accumulated strain energy density
- Incremental plastic free energy density
- Accumulated plastic free energy density
• Incremental plastic dissipation density

• Accumulated plastic dissipation density

• The last 4 slots are currently empty, reserved for potential future use

Note that "incremental" means the change of energy density during the time step, "accumulated" means current cumulative energy density at the time step.

Here is an example showing how to obtain the energy density evolution of a specific element. Say you are interested in looking at the accumulated plastic dissipation density of an 8NodeBrick element with element tag 73. Go to Index_to_Energy_Element, find the index for element 73 to be 612. Since accumulated plastic dissipation density is the 8th entry among the 12 slots, the row index for the data of your interest is $612 + (8-1) = 619$. This means that the accumulated plastic dissipation density of your chosen element is stored at row 619 of Energy_Density_Element_Average.

### 206.9.4 Energy Quantity of Element

Energy_Element dataset stores averaged energy density quantities of each element. To find the energy time history of a specific element, find the index in the Index_to_Energy_Element dataset then find the corresponding entry in the Energy_Element dataset.

For each element, 12 data slots are occupied in the following order:

• Incremental kinetic energy

• Accumulated kinetic energy

• Incremental strain energy

• Accumulated strain energy

• Incremental plastic free energy

• Accumulated plastic free energy

• Incremental plastic dissipation

• Accumulated plastic dissipation

• Incremental viscous energy dissipation

• Accumulated viscous energy dissipation

• The last 2 slots are currently empty, reserved for potential future use
Note that "incremental" means the change of energy density during the time step, "accumulated" means current cumulative energy density at the time step.

Here is an example showing how to obtain the energy evolution of a specific element. Say you are interested in looking at the accumulated plastic dissipation of an 8NodeBrick element with element tag 73. Go to Index_to_Energy_Element, find the index for element 73 to be 612. Since accumulated plastic dissipation is the 8th entry among the 12 slots, the row index for the data of your interest is $612 + (8-1) = 619$. This means that the accumulated plastic dissipation of your chosen element is stored at row 619 of Energy_Element.
Chapter 207

Real-ESSI Pre Processing and Model Development Methods


(In collaboration with Mr. Sumeet Kumar Sinha, Dr. Hexiang Wang and Dr. Yuan Feng)
207.1 Introduction

207.2 Model Development Using gmsh

207.2.1 Introduction to gmESSI

The gmESSI, pronounced as [gm-ESSI], is a translator that converts mesh file from gmsh (a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities) to Real-ESSI DSL format. The primary aim of this program is to provide an efficient pre-processing tool to develop Finite Element (FE) models in gmsh and make them interface with various Real-ESSI functionalities. The gmESSI translator package contains the translator, sublime plugin and the manual.

The gmESSI package is available at http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/_Chapter_SoftwareHardware_Pre_Processing/Real-ESSI_gmESSI.tgz.

The text editor sublime plugin [gmESSI-Tools] can be downloaded here: http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/_Chapter_SoftwareHardware_Pre_Processing/fei-syntax-n-snippets.tar.gz.

207.2.1.1 Getting Started

The translator utilizes the physical and entity group concept of Gmsh (http://geuz.org/gmsh/doc/ texinfo/gmsh.html) (Geuzaine and Remacle, 2009), which gets imprinted in the mesh "msh" file. The translator then manipulates these groups to convert the whole mesh to ESSI commands. Thus, making physical groups is the essential, key for conversion. The Translator basically provides some strict syntax for naming these Physical Groups which provides gmESSI information about the elements or (nodes) on which the translation operates. The translator is made so general that any other FEM program can use it with little tweaks to have their own conversion tool. A quick look at some important features of the program are:

- It has a lot of predefined commands which do the conversion at the blink of an eye. These commands make it easier to define elements, boundary conditions, contacts/interfaces, fixities, loads ....

- It provides a python module "gmessi". The users can import this module and can extend the functional capability of gmESSI.

- The [gmESSI-Tools] sublime plugin makes it easy by providing syntax coloring and auto-text-completion for gmESSI commands. [gmsh-Tools] sublime plugin can also be installed for gmsh syntax coloring and auto-completion.
• The translator uses a *mapping.fei* file to check for its command syntax and conversion. A user can easily add a command in *mapping.fei* and it would get reflected automatically in the translation.

• It automatically optimizes the Real-ESSI tags (node, element, load) for space and time efficiency while running simulation.

**Installation Process:** The Translator have its dependencies on Octave (3.2 or higher), Boost(1.58 or higher), (Python 2.7 or higher). One should make sure to have them before compiling it. On Linux Ubuntu distros the dependencies can be installed as

```
sudo apt-get install liboctave-dev
# Boost version should be higher than 1.48
sudo apt-get install libboost-all-dev
sudo apt-get install python-dev
```

Installation of the gmESSI translator is easy, just follow steps below.

```
## go to folder where you want to store and build gmESSI application
## download the package from main Real-ESSI repository
## this line below should be all one line
## HOWEVER it had to be broken in two lines to be readable
## so please make a single command out of two lines below
##
#
# using curly brackets to help in checking scripts, that rely on these
# brackets being available around URL
#
gwget {http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/
   _Chapter_SoftwareHardware_Pre_Processing/Real-ESSI_gmESSI.tgz}
## make directory, move files, expand archive
mkdir Real-ESSI-gmESSI
mv _all_files_gmESSI_.tgz Real-ESSI-gmESSI
cd Real-ESSI-gmESSI
tar -xvzf _all_files_gmESSI_.tgz
## build the package
make # builds the application in curr_dir/build
## install the package
# -- by default the package is installed in /usr/local
make install # installs the package in /usr/local
# -- to change the install directory
make install INSTALL_DIR=install_dir_specified_by_user
```

# For installation of gmESSI plugin in sublime
```
# open sublime-text
```
207.2.1.2 Running gmESSI

gmESSI can be invoked from the bash terminal by typing `gmessy`. It can take one or multiple `xyz.gmessi` files as an argument and convert them to Real-ESSI files in their respective simulation directory defined by the user. By default, the 'gmessi' python module is automatically imported and available as 'gmESSI' in '.gmeesi' input file.

`gmessy` is a top level python script that parses the .gmessi file and categories commands in the following order as

- **gmESSI Command**: gmESSI Commands are one line commands. They start and end with ‘[’ and ‘]’ respectively. Section 207.2.3 describes the syntax.

- **gmESSI Comments**: The lines that start with ‘//’ are considered as gmESSI comments. It gets translated and copied to the main file (See Section 207.2.4.6).

- **Singular Commands**: The lines that start with ‘!’ are directly copied to the main file (See Section 207.2.4.6 and Section 207.2.5.1). Real-ESSI domain specific language (DSL) are written following the exclamation mark ‘!’ sign.

- **Python Comments**: The lines that start with ‘#’ are considered as python comments.

- **Python Commands**: Whatever lines left are considered as python commands. This option is only for the advanced user and is not documented to make the manual simple. Only some useful commands required are explained in the manual.

The categorized commands then generates an equivalent python (.py) script, which gets finally run in python interpreter. The generated equivalent python script can be seen by adding ‘-l’ or ‘–logfile=LOG_FILE’ option during execution. It is important to note that nodes, coordinates, element no etc generated from the translator have a precision associated with them. By default the precision is up-to ‘6’ significant digits. The user can change the precision anywhere in the .gmessi file as
gmESSI.setPrecision(10);

This will set the precision to ‘10’ significant digits. Lowering precision can be helpful in generating same coordinates for contact/interface node pairs. See [Example_4.gmessi] for its usage.

The full description of gmessy can be invoked from the terminal as

```
$gmessy --help
```

```
     [-nodegid= NM_FILE] [-elegid= ELM_FILE]  
gmessi_filename

positional arguments:  
gmessi_filename filename containing semantics of conversion

optional arguments:  
-h, --help show this help message and exit  
-l generate the log file at the current location  
-nm generate the node map file at the current location  
-em generate the element map file at the current location  
-ne don't carry out the conversion  
--logfile= LOG_FILE generate the log file at specified location  
--nodegid= NM_FILE generate the node-map (gmsh-to-Real_ESSI) file at specified location  
--elegid= ELM_FILE generate the element-map (gmsh-to-Real_ESSI) file at specified location
```

Since gmESSI optimizes the ‘node’ and ‘element’ tag for Real-ESSI, it provides an interface to retrieve the node map and element map containing mapping from gmsh.tag to Real_ESSI.Tag.

Running gmESSI requires, the .gmessi input file and the gmsh mesh (.msh) mesh file containing physical groups. Let’s go and run an example to see how gmESSI works. [Example_1] can be obtained here.

Alternatively in the gmESSI directory, navigate to the Examples directory and then to Example_1 directory.

```
$cd ./Examples/Example_1
$ls
Example_1.geo # geometry file [gmsh]  
Example_1.msh # mesh file [gmsh]  
Example_1.gmessi # gmessi input file [gmESSI]
```

Contents of Example_1.gmessi input file: As described above, the .gmessi input file contains gmESSI commands, singular commands and python commands. Also, it can contain comments followed by //
or #. At the beginning of the input file, the simulation directory, main, node, element, load file-names must be specified. Also, before adding any gmESSI command, mesh must be loaded using ‘gmESSI.loadGmshFile’ command.

```plaintext
$ cat Example_1.gmessi

### loading the msh file
gmESSI.loadGmshFile("Example_1.msh")

### Physical Groups defined in the msh file.
#2 2 "Base_Surface"
#2 3 "Top_Surface"
#3 1 "Soil"

### Defining the Simulation Directory
gmESSI.setSimulationDir("./Example_1_ESSI_Simulation")
gmESSI.setMainFile(gmESSI.SimulationDir+ "main.fei")
gmESSI.setNodeFile(gmESSI.SimulationDir+ "node.fei")
gmESSI.setElementFile(gmESSI.SimulationDir+ "element.fei")
gmESSI.setLoadFile(gmESSI.SimulationDir+ "load.fei")

// My new model
! model name "Soil_Block";
[Add_All_Node{ unit:= m, nof_dofs:= 3}]

// Adding Material layer wise and also assigning it to elements
[Vary_Linear_Elastic_Isotropic_3D{Physical_Group#Soil, ElementCommand:=←
  [Add_8NodeBrick{}], Density:= 1600+10*(10-z)
  0 \ kg/m^3, ElasticModulus:= 20e9+10e8*(10-z)-8\Pa, PoissonRatio:= 0.3}]

! include "node.fei";
! include "element.fei";
! new loading stage "Stage1_Surface_Loading";

# Applying Fixities
[Fix_Dofs{Physical_Group#Base_Surface, all}]

### For applying Surface load on the Top Surface of the Soil Block
[#[Add_8NodeBrick_SurfaceLoad{Physical_Group#1,Physical_Group#3,10*Pa}]]

### For applying Nodal loads to all the nodes of the top surface
[Add_Node_Load_Linear{Physical_Group#Top_Surface, ForceType:= Fx, Mag:= 10*kN}]

### For applying Self-Weight Load to the soil elements
! add acceleration field # 1 ax = 0*g ay = 0*g az = -1*g ;
! add load #18 to all elements type self_weight use acceleration field # 1;

# Updating the tag inside gmESSI as user entered by himself load tag
gmESSI.setESSITag("load",19)
```
Running Example 1 in Terminal:

```plaintext
$ gmessy Example_1.gmessi
Message:: newDirectory created as ./Example_1_ESSI_Simulation

Add_All_Node{ unit:= m, nof_dofs:= 3}
  Found!!
  Successfully Converted

Vary_Linear_Elastic_Isotropic_3D{Physical_Group#Soil, ElementCommand:= [Add_8NodeBrick{}], Density:= 1600+10*(10-z) \ 0 \ kg/m^3, ElasticModulus:= 20e9+10e8*(10-z)\-8\Pa, PoissonRatio:= 0.3}
  Found!!
  Successfully Converted

Fix_Dofs{Physical_Group#Base_Surface, all}
  Found!!
  Successfully Converted

Add_Node_Load_Linear{Physical_Group#Top_Surface, ForceType:= Fx, Mag:= 10*kN}
  Found!!
  Successfully Converted

*************** Updated New Tag Numbering **********************
damping = 1
displacement = 1
element = 28
field = 1
load = 17
material = 4
motion = 19
node = 65
nodes = 65
Gmsh_Elements = 46
Gmsh_Nodes = 65
```

It must be noted that the terminal only displays information about gmESSI commands. The singular commands are directly copied to the main file. The translator creates a user defined directory `Example_1_ESSI_Simulation` and places...
1. node.fei
2. element.fei
3. load.fei
4. main.fei
5. Example_1.msh

The terminal displays the WARNING, ERROR messages and log of command conversions as shown above. At the end, it displays the Available ESSITag’s numbering, which can be refereed and used for further conversion.

ESSITags are explained later in this manual in Section 207.2.4.8.

The Real-ESSI input files produced can be tweaked a little if required. Once all is set, the model can be run through Real-ESSI Simulator

```bash
1 cd Example_1_ESSI_Simulation
2
3 ### To run ESSI in sequential
4 # -- assuming sequential executable name is 'essi'
5 essi -f main.fei
6
7 ### To run ESSI in parallel
8 # -- assuming parallel executable name is 'pessi'
9 mpirun -np 4 pessi -f main.fei
```
207.2.2 Gmsh Physical Groups and Geometrical Entities

207.2.2.1 Geometrical Entities

Geometrical entities are the most elementary group in Gmsh. Each point, line, surface and volume is a geometrical entity and possess a unique identification number. Elementary geometrical entities can then be manipulated in various ways, for example using the Translate, Rotate, Scale or Symmetry commands. They can be deleted with the Delete command, provided that no higher-dimension entity references them. Example 2.geo shows description of a geometry (.geo) file in gmsh for creating a cantilever beam. The files can be downloaded here. Alternatively, it can be located in the gmESSI directory by navigating to the Examples/Example 2 directory.

```bash
$ cat Example_2.geo

// Creating a point
Point(1) = {0,0,0};

// Dividing the beam length in 5 parts
Extrude (4,0,0) {Point{1}; Layers{5};}

// Dividing the beam width in 2 parts
Extrude (0,1,0) {Line{1}; Layers{2};Recombine;}

// Dividing the beam depth in 2 parts
Extrude (0,0,1) {Surface{5}; Layers{2};Recombine;}
```

Figure 207.1 shows, the different unique identification number attached to each of the nodes, lines, surface and volume of the geometry of cantilever beam. Physical groups can now be created of type {nodes, lines, surface or volume} containing one or more geometrical entities of their respective type.

207.2.2.2 Physical Groups

Physical groups are groups of same type {nodes, lines, surface, volume} of elementary geometrical entities. These Physical Groups cannot be modified by geometry commands. Their only purpose is to assemble elementary entities into larger groups, possibly modifying their orientation, so that they can be referred to by the mesh module as single entities. As is the case with elementary entities, each physical point, physical line, physical surface or physical volume are also assigned a unique identification number.

**NOTE:** A geometrical entity has only one elementary entity number but can be a part of many physical groups by sharing their unique identification number.

Below is the continuation of Example 2.geo in Gmsh for creating physical Groups of cantilever beam.
(a) All Points
(b) All Lines
(c) All Surface
(d) All Volume

Figure 207.1: Showing Geometrical Entities. Every point, line, surface and volume has an unique identification number assigned to it.

Just for the sake of example, 4 physical groups are created which consist of all points, lines, surface and volume respectively of the cantilever beam model. Also physical groups of the surface where fixities and load is applied is created.

```
$ cat Example_2.geo
.....
Physical Point ("All_Points") ={1,2,3,4,5,6,10,14};
Physical Surface("All_Surfaces") = {5,14,22,27,18,26};
Physical Line("All_Lines") ={1,2,3,4,12,13,21,17,7,8,9,10};
Physical Volume("All_Volumes") ={1};
Physical Surface("ApplySurfaceLoad") ={27};
Physical Surface("SurfaceToBeFixed") ={26};
```

In generated mesh (.msh) file, all the geometrical entities have a tag list which contains the ids of the physical groups to which it belongs or is associated. In the above example shown in Figure 207.2, every point, line, surface, volume belongs to only one physical group and thus are showing only one associative number against themselves. Figure 207.3 shows geometrical entities which are part of many physical groups. For example:- the volume shown in Figure 207.3 shows physical group of volumes having id 1 and 7.

The whole idea of creating a Physical Group of points, lines, surfaces and volumes and giving it a unique string name is to allow quick identification and manipulation during gmESSI commands. In Gmsh the
(a) Physical group of Points

(b) Physical group of Lines

(c) Physical group of Surface

(d) Physical group of Volume

Figure 207.2: Showing all 4 Physical Groups with entities numbered by their physical group id’s.

Figure 207.3: Showing geometrical entities associated with more than one physical group.

The name of these Physical Group along with their corresponding elements and nodes gets transferred to the mesh .msh file as shown below. Figure 207.4 shows how Gmsh interprets these Physical groups in .msh file.

```
$cat Example_1.msh
.....
$PhysicalNames
6
0 1 "All_Points"
1 3 "All_Lines"
2 2 "All_Surfaces"
```
NOTE: While creating a physical group in Gmsh, only the information (nodes and elements) of that physical group gets written in the .msh file and rest are not written. So one must be careful to create physical groups of all entities which is needed during post-processing or conversion. More information about Gmsh syntax, physical groups, commands, .msh file, save options, is available at the main online documentation web site: http://geuz.org/gmsh/doc/texinfo/gmsh.html

![Figure 207.4: Description showing how gmsh interprets the Physical Groups.](image)

- **Physical Group Description ::** Gmsh uses it to identify the type of physical group. 0, 1, 2 and 3 represents the physical group of geometric points, lines, surface and volume respectively.

- **Physical Group Unique Identification Number ::** It is an unique identification number automatically assigned to each physical group by gmsh.

- **Physical Group Unique Name ::** It is also the same as Physical Group Unique Identification Number but the difference is that it is not automatic but defined by the user and that too in the form of string.

The gmESSI Translator utilizes the property of naming the physical group as "string" to get gmESSI commands from the user along with specific physical group on which it is operated. Below in shown [Example_2.gmessi] input file for a Cantilever analysis. It shows how to write gmESSI commands with physical group information on which it is operated. gmESSI utilizes the mesh (.msh) file to get the respective physical group and translated it to ESSI input (.fei) files.

```bash
$ cat Example_2.gmessi

3 gmESSI.loadGmshFile("Example_2.msh")
```
### Physical Groups Available in Example_2.msh file

#0 1 "All_Points"
#1 3 "All_Lines"
#2 2 "All_Surfaces"
#2 5 "ApplySurfaceLoad"
#2 6 "SurfaceToBeFixed"
#3 4 "All_Volumes"

############ Important!! to set the file names ###############

```python
gmESSI.setSimulationDir("./Example_2_ESSI_Simulation")
gmESSI.setMainFile(gmESSI.SimulationDir+ "main.fei")
gmESSI.setNodeFile(gmESSI.SimulationDir+ "node.fei")
gmESSI.setElementFile(gmESSI.SimulationDir+ "element.fei")
gmESSI.setLoadFile(gmESSI.SimulationDir+ "load.fei")
```

// My new model
```plaintext`
! model name "Cantilever_Analysis";

[Add_All_Node{Unit:= m, NumDofs:= 3}]

// Adding Material
```plaintext`
! add material 1 type linear_elastic_isotropic_3d mass_density = 2000*kg/m^3 ➩
elastic_modulus = 200*MPa poisson_ratio = 0.2;
```

```plaintext`
[Add_8NodeBrick{Physical_Group#All_Volumes, material_no:= 1}]
[Fix_Dofs{Physical_Group#SurfaceToBeFixed, all}]
```

! include "node.fei";
! include "element.fei";

! new loading stage "Stage1_Uniform_Surface_Load";

# Adding Surface Load
```plaintext`
#[Add_8NodeBrick_SurfaceLoad{Physical_Group#All_Volumes, ➩
Physical_Group#ApplySurfaceLoad, -10*Pa}]
[Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, ForceType:= Fz, Mag:= ➩
-10*kN}]
```

! include "load.fei"
! define algorithm With_no_convergence_check;
! define solver UMFPack;
! define load factor increment 1;
! simulate 10 steps using static algorithm;
! bye;
```

NOTE:- The first command in [.gmessi] file should be to load the mesh (.msh) file. The syntax to load the gmsh generated mesh file is
```python
gmESSI.LoadGmshFile("meshfile.msh")
```
The gmESSI translator reads the command \[\text{Add.All.Node\{ Unit:= m, NumDofs:= 3\}}\] and adds all the nodes from mesh file to ESSI input files. Similarly it translates all the other commands as well.
207.2.3 gmESSI Command Description

gmESSI Translator as said above utilizes the naming of the physical groups to get commands from the user and then carry out the conversion by acting on the defined physical group.

207.2.3.1 gmESSI Syntax

gmESSI follows strict syntax. gmESSI parses the physical group name string in mesh (.msh) file. Let us have a quick look at the syntax of physical group name.

**Physical Group Names:** Physical group names are created inside gmsh geometry file. gmESSI follows special syntax as described below.

1. Physical group names used in gmsh should be unique for gmESSI to identify them during post processing.
2. Physical group names should not contain any space
3. Physical group tags can be any alphanumeric sequence but should not contain any of these [ ] $ literals in their names. Example "Physical_Group_1"

**gmESSI Command Syntax:** gmESSI translator commands are always enclosed between opening/-closing square brackets [ and ] respectively. A typical gmESSI command syntax is shown in Figure 207.5

```
[Add_Node_Load_Linear{Physical_Group#ApplyLoad, ForceType:= Fx, Load:= 10*kN}]
```

Figure 207.5: gmESSI command description.

- **Command Name:** Just as regular function gmESSI Commands have a name and take arguments. The names are usually self explanatory of its function like *Add_8NodeBrick*{...}, *Free_Dofs*{...} .. etc

- **Physical Group Argument:** Usually the gmESSI commands have first argument as physical group. For Example:- *Add_8NodeBrick*{PhysicalGroup#5,...}, *Add_8NodeBrick*{PhysicalGroup#All_Volumes,...}, *Free_Dofs*{PhysicalGroup#4,...},.. etc.

  *Physical Group Id* can be the gmsh unique string or number representing that physical group (as shown in .msh file).
• **Arguments**: Arguments as always are separated by comma ",".

  – **Argument Tag** The arguments of gmESSI commands can also have tags associated with them so that it becomes easy for the user to interpret the argument and make changes in future. The tag and the argument is separated by `:=`. Tag itself has no meaning but it serves as an important information center for user. An example is shown below to show how tags are applied.

  – gmESSI command having arguments without tags
    1. `[Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, Fz, -10*kN}]`

  – gmESSI command having arguments with tags
    1. `[Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, ForceType:= Fz, Mag:= -10*kN}]`
    2. `[Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, ForceType:= Fz, -10*kN}]`
    3. `[Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, Force_Direction:= Fz, Strength:= -10*kN}]`

    It can be seen from above examples that the tags are optional and also the user can put their own tag names. The sublime plugin [gmESSI-Tools] comes with elaborative tags for the parameters and a lot more with syntax coloring and text-completion for gmESSI commands. It is encouraged to use the plugin and take its advantage.

![gmESSI conversion description](image)

---

**Figure 207.6**: gmESSI conversion description.
Figure 207.6 shows the illustration how gmESSI works. Load gets added to all the nodes of the physical group ‘ApplyLoad’. gmESSI translator automatically assigns the unique load tag sequentially. It retrieves the node tag from the physical group. Rest of the information like ‘ForceType’ and ‘Magnitude’ is obtained from the arguments.

Most of the time these arguments are dummy which means that they just get copied to their equivalent ESSI command at their respective places. These arguments thus have a "string" data-type. For example: the command Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, Fz, -10*kN} is equivalent to the Real-ESI command add load #{} to node #{} type linear {} = {}. Fz and -10*kN goes to their respective position directly through the translator as shown in the Figure 207.6 load number 1 and node number 32 are computed by the translator and then inserted in the ESSI command.

NOTE:- The gmESSI Translator does not provide syntax checking for those dummy arguments. It means that, whatever is written gets copied at the respective position in the equivalent ESSI command, so the one must be careful with what they are writing in these arguments. For Example the command Add_Node_Load_Linear{Physical_Group#Id, ForceDirection, Magnitude} based on the arguments can get converted as

1. [Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, ForceType:= Fz, Mag:= -10*kN}]  
   -- > add load #1 to node #32 type linear Fz = -10*kN  
   -- > add load #2 to node #33 type linear Fz = -10*kN  
   ......  
   -- > add load #100 to node #100 type linear Fz = -10*kN

2. [Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, ForceType:= Fz, Mag:= -10}]  
   -- > add load #1 to node #32 type linear Fz = -10  
   -- > add load #2 to node #33 type linear Fz = -10  
   ......  
   -- > add load #100 to node #100 type linear Fz = -10

3. [Add_Node_Load_Linear{Physical_Group#ApplySurfaceLoad, ForceType:= Ft, Mag:= -10*kN}]  
   -- > add load #1 to node #32 type linear Ft = -10*kN  
   -- > add load #2 to node #33 type linear Ft = -10*kN  
   ......  
   -- > add load #100 to node #100 type linear Ft = -10*kN

All the above conversions are correct. But only conversion (1.) is correct as an input for Real-ESI
Simulator because force direction is one of $F_x, F_y, F_z$ and magnitude $10^4 kN$ has proper units. So one must be very careful while writing the arguments.

**Note:** Some of the arguments are not string but represents numerical quantities, which are manipulated by the translator during conversion. Thus, the one must supply only numbers without any alphabets else it would lead an unexpected termination of program. These arguments corresponds to *Special Commands* such as *Connect Command* and *Variational Commands*. The manual talks about them later in Section 207.2.5.7.

### 207.2.3.2 gmESSI Command’s Physical Group

As iterated earlier, gmESSI commands operates on physical groups. The gmESSI command usually have their first argument as physical on which it operates. The gmESSI syntax allows the users to operates it’s command on specific physical groups. The user specifies the group by including an argument Physical_Group#Tag in front of the gmESSI commands describing the command. The tag can be either Physical_Group_Id, Physical_Group_Name. Let’s look at some of them

- [Add_Node_Load_Linear{Physical_Group#5,Fz,-10*kN}] operates on physical group 5
- [Add_8NodeBrick{Physical_Group#All_Volumes, 1}] operates on physical group which has string tag as All_Volumes

For example in reference to [Example_2.gmessi] Physical_Group#All_Volumes or Physical_Group#4 refers the same physical group.

A physical group is a group of point, line, surface or volume defined by the user which contains all the geometrical entities that falls under that domain/group. Figure 207.2 shows physical groups.
207.2.4 gmESSI Output

gmESSI Translator translates the gmESSI commands operated on mesh (.msh) file to different ESSI input (node, element, load and main) (.fei) files and put them in user-defined directory. It also updates the mesh (.msh) file and puts it in the same directory. The log of translation, errors and warnings are displayed on the terminal. Below is the demonstration of log messages one by one using [Example_2.gmessi] with mesh-file name Example_2.msh. The folders and Reall ESSI input (.fei) files that are created by the translator for Example_2.gmessi input file are.

207.2.4.1 Directory Example_2_ESSI_Simulation

gmESSI Translator creates simulation directory as specified by the user. The user is expected to create the necessary node (Section 207.2.4.4), element (Section 207.2.4.3), load (Section 207.2.4.5) and main (Section 207.2.4.6) file to that directory. The user is expected to provide the directory and filenames before executing any gmESSI command. In case the directory already exists a warning messages is shown on the terminal and a new directory following the original name with ‘_n’ (n is number) is created. A new Real-ESSI simulation directory is assigned by the following command

```
gmESSI.setSimulationDir("./Example_2_ESSI_Simulation", overwrite\_mode)
```

where, 'overwrite_mode=0' means that in case of already existing folder, a new directory following the original name with ‘_n’ (n is number) is created. 'overwrite_mode=1' would not check for any conflicts and use the same directory as specified by user. For example: running [Example_.gmessi] file would produce the following message.

```
$ gmessy Example_2.gmessi
Files converted to Examples/Example_2_ESSI_Simulation
```

Again, running the same example would produce the following message as shown below. In [Example_2.gmessi] overwrite is turned off and thats why it creates new-non conflicting directory by appending 1 to end.

```
$ gmessy Example_2.gmessi
Message:: newDirectory created as ./Example_2_ESSI_Simulation_1
```

The execution of gmessy XYZ.gmessi produces warnings/errors in the following situations.

- **ERROR:: Please Enter the gmessi File ::** It occurs if the user does not give a filename. The possible situation for getting this error is

```
$ gmessy
```

• **ERROR::** The program failed to open the file XYZ.msh It occurs if the given file or one of the files in the argument does not exist or fails to open because of some reason.

• **WARNING::** Directory Already Present. The contents of the Folder may get changed :: It occurs when users translates the mesh file file XYZ.msh in overwrite mode and the corresponding folder XYZ_ESSI_Simulation already exists at the execution location.

• **Files converted to Examples/Example_2_ESSI_Simulation ::** The message refers to the location of the folder where the translations have been saved.

### 207.2.4.2 Translation Log Terminal

gmESSI Translator displays the log of translation of gmESSI commands to corresponding *Real-ESSI commands* on the terminal. Proper *Errors Messages* and *Warnings* are echoed to the user. The execution of the commands are sequential which means the commands written first are executed first and similarly their success and failure is also echoed first. Let us look at this aspect with Example_2.gmessi.

```
$ cat Example_2.gmessi
......
! add material 1 type linear_elastic_isotropic_3d mass_density = 2000*kg/m^3 ←
   elastic_modulus = 200*MPa poisson_ratio = 0.2;

[Add_8NodeBrick{Physical_Group#All_Volumes, material_no:= 1}]

[Fix_Dofs{Physical_Group#SurfaceToBeFixed, all}]

! include "node.fei";
! include "element.fei";
......
```

Here, the sequence of execution of commands is `'! add material # 1 type linear_elastic_isotropic_3d mass_density = 2000 * kg/m^3 elastic_modulus = 200 * MPa poisson_ratio = 0.2; ', [Add_8NodeBrick{ Physical_Group#All_Volumes, material_no:= 1}]. [Fix_Dofs{ Physical_Group#SurfaceToBeFixed, all}] and `'! include “node.fei”;'`. Notice that the same order gets reflected in the translation log on the terminal as shown below. Also, it must be noted that the commands followed by `'!'` or `'//'` or `'#'` or python commands do not have any log messages corresponding to them.

It must be noted that the lines following `'!'` are directly copied to the main (Section 207.2.4.6). Usually Real-ESSI domain specific language that does not operate/require any physical group should be written following exclamation `'!'` sign.

```
$ gmessy ./Example_2.gmessi
......
```
Apart from displaying the log details on the terminal, similar log is added for each translation of gmESSI commands in their respective files in which they are translated. In these files, each successful translation is enclosed between corresponding `RespectiveGmESSICommand Begins` and `RespectiveGmESSICommand Ends`. The same is shown below through the contents of node.fei. Notice that all the translations are enclosed between Begins and Ends Tag.

```plaintext
$ cat Examples/Example_2_ESSI_Simulation/node.fei

//****************************************************************
// Add_All_Node{Unit:= m, NumDofs:= 3}Starts
//****************************************************************

add node # 1 at (0.000000*m,0.000000*m,0.000000*m) with 3 dofs;
add node # 2 at (4.000000*m,0.000000*m,0.000000*m) with 3 dofs;
add node # 3 at (0.000000*m,1.000000*m,0.000000*m) with 3 dofs;

Adresselement...nings

//****************************************************************
// Add_All_Node{Unit:= m, NumDofs:= 3}Ends
//****************************************************************

$ cat Examples/Example_2_ESSI_Simulation/element.fei

//***********************************************************************
// Add_8NodeBrick{Physical_Group#All_Volumes, material_no:= 1}Starts
//***********************************************************************

add element #1 type 8NodeBrick with nodes (51,46,29,37,33,17,1,9) use material #1;
add element #2 type 8NodeBrick with nodes (47,28,5,19,51,46,29,37) use material #1;
add element #3 type 8NodeBrick with nodes (42,32,46,51,13,3,17,33) use material #1;

Adresselement...nings

//***********************************************************************
// Add_8NodeBrick{Physical_Group#All_Volumes, material_no:= 1}Ends
//***********************************************************************

NOTE: The ordering/sequence of commands in ESSI analysis file is important and so the user must
make sure that the translations are made in the same order or if not the user should change it manually by (cut/copy/paste) in node.fei, load.fei and main.fei files before execution.

Having given a short description of the other translation log/error messages. Let us look more closely one by one and understand the messages, errors and warnings prompted on the terminal.

- **Found!!**: This message in front of the gmESSI command as shown above on translation log in the terminal means that, the corresponding command was found in the gmESSI Command Library.

- **Successfully Converted**: As the message itself describes, it occurs if the command has been successfully translated.

- **Not Found!!**: It occurs if the gmESSI Translator could not find the arbitrary command XYZ in the gmESSI Command library. Example:- Loading\{Fx,10*kN\} NotFound!!

- **WARNING:: Execution of the command escaped. The Gmessi command XYZ could not be found**: The gmESSI Translator does not terminate the translation if a command is not found, instead gives this warning message following the Not Found!! Error.

- **Error:: The command XYZ has a syntax error in Physical_Group# tag**: It occurs if there is a syntax error in Physical_Group# argument. The correct representation for Physical group Tags is Physical_Group#n, where n is the group id as 1,2,3.. etc. Examples of improper representation are Phy#2, Physical#Node,..

- **Warning:: The command XYZ failed to convert as there is no such Physical Group**: It occurs if one of the arguments in the command is Physical_Group# and the specified physical group by the user does not exist in the .msh file.

- **Warning:: The command XYZ could not find any nodes/elements on which it operates**: It occurs if for a specified command, the required element types for translation could not be found in the specified Physical group. For Examples:- [Add_8NodeBrick{Physical_Group#1,1}] would give this warning as the Physical_Group#1 being a Physical line group does not contain any 8-Noded Brick elements on which this command operates.

- **ERROR:: Gmsh File has invalid symbols in Node Section. Unable to convert string to integer in Gmsh File**: It occurs if there is perhaps a string inside the Nodes section of .msh file.

- **ERROR:: The command XYZ has a syntax errors**: It occurs if the specified command by the user contain any syntax errors caught while parsing the command.
• ERROR:: Gmsh File has invalid symbols in Element Section. Unable to convert string to integer in Gmsh File: It occurs if there is perhaps a string inside the Element section of .msh file.

207.2.4.3 Element File (element.fei)

Element file element.fei is one of four parts of Real-ESSI input file that contains the translation of commands related to only initialization of elements of the FEM mesh. Generally, all the conversions from Elemental Command (Section 207.2.5.5) are written to element file.

A new analysis element file is assigned by the following python command

```
    gmESSI.setElementFile(gmESSI.SimulationDir+ "element.fei")
```

where, ‘gmESSI.SimulationDir’ returns the Real-ESSI Simulation directory specified by the user (see section 207.2.4.1).

207.2.4.4 Node File (node.fei)

Node file node.fei is one of four parts of Real-ESSI input file that contains the translation of commands related to only initialization of nodes of the FEM mesh. All the conversions from Add Node Command (Section 207.2.5.2) are written to node file.

A new analysis node file is assigned by the following python command

```
    gmESSI.setNodeFile(gmESSI.SimulationDir+ "node.fei")
```

where, ‘gmESSI.SimulationDir’ returns the Real-ESSI Simulation directory specified by the user (see section 207.2.4.1).

207.2.4.5 Load File (load.fei)

Load file load.fei contains the translation of commands related to the load and boundary conditions on the structure, for example declaration of fixities, boundary conditions, tied/connected nodes, nodal loads, surface loads etc....

A new load file is assigned by the following python command

```
    gmESSI.setLoadFile(gmESSI.SimulationDir+ "load.fei")
```

where, ‘gmESSI.SimulationDir’ returns the Real-ESSI Simulation directory specified by the user (see section 207.2.4.1).
207.2.4.6 Analysis File (main.fei)

Analysis file `main.fei` is the main file which is run on Real-ESSI Simulator. The main file must include load, node and element file through `include 'filename.fei'` command.

A new analysis main file is assigned by the following python command

```python
gmESSI.setMainFile(gmESSI.SimulationDir+ "main.fei")
```

where, `gmESSI.SimulationDir` returns the Real-ESSI Simulation directory specified by the user (see section 207.2.4.1). A typical analysis file after conversion looks like the following.

```fei
$ cat Examples/Example_2_ESSI_Simulation/Example_2_analysis.fei
// My new model
model name "Cantilever_Analysis";

// Adding Material
add material 1 type linear_elastic_isotropic_3d mass_density = 2000*kg/m^3 ←
elastic_modulus = 200*MPa poisson_ratio = 0.2;

include "node.fei";
include "element.fei";

new loading stage "Stage1_Uniform_Surface_Load";

include "load.fei";
define algorithm With_no_convergence_check;
define solver UMFPack;
define load factor increment 1;
simulate 10 steps using static algorithm;
bye;
```

The user can now add solver, time steps and even rearrange the file structure accordingly to Real-ESSI syntax.

**NOTE:** Real-ESSI Interpreter is sequential and follows certain ordering in commands like materials should be declared before assigning to elements, main-follower nodes can be assigned only when both nodes are declared .. etc.. One should be careful with the order in which conversions are made and if necessary should change it manually by (cut/copy/paste) later in the files geometry.fei, load.fei and analysis.fei or use the python module discussed later before running in ESSI. Please refer to the Real-ESSI manual for more details on the ordering of the commands.
207.2.4.7 Mesh File (XYZ.msh)

Mesh file XYZ.msh is the input required by the translator. The translator updates the mesh file with users addition. For example:- if Connect-Command (Section 207.2.5.8) is used, the file contains additional physical group, nodes and 2-noded elements. The Connect Command is discussed in the more detail later in Section 207.2.5.8.

207.2.4.8 Updated ESSI Tags Terminal

Updated ESSI Tags refers to the new tag numbering reference associated with ESSI Tags. ESSI has tag numberings associated for damping, displacement, element, field, load, material, and node/nodes. For example in Real-ESSI Command `add node # 1 at (x,y,z) with 3 dofs`, node is a tag and requires a new number like 1 to be associated with that node. The translator displays the new numberings available for each ESSI Tag so that the user is made aware of new numberings for manually specifying an ESSI command after the translation.

gmESSI also provides a python command to set the ESSI Tag. The command is

```python
1 gmESSI.setESSITag(ESSI_Tag_Name, Tag)
```

where,

- **ESSI_Tag_Name**: It refers to a string representing to the Real-ESSI tag such as ‘node’, ‘element’, ‘field’...etc

- **Tag**: It refers to an integer representing the next available tag.

**NOTE**: If user is writing its own Real-ESSI domain specific language (DSL), it is expected that the user will update the corresponding Real-ESSI tag used in that DSL. Otherwise, gmESSI would not be able to know the updated available tags. See Example_1.gmessi for its usage.

```bash
1 $ gmessy Example_2.gmessi
2 ..................  
3 ******* Updated New Tag Numbering *******
4 Damping = 1
5 displacement = 1
6 element = 21
7 field = 1
8 load = 19
9 material = 2
10 motion = 1
11 node = 55
12 nodes = 55
13 Gmsh_Elements = 127
14 Gmsh_Nodes = 55
```
207.2.5 gmESSI Commands

Having the knowledge about the syntax, output files, errors and warnings, its time to move on to different types of commands that gmESSI offers. It provide commands operated on physical group to allow conversion for to equivalent Real-ESSI commands. There are also some special command that gmESSI supports. For simplicity, the commands are categorized on the basis of their operation on nodes/elements. As stated earlier, the commands are translated to one of the four files node.fei, element.fei, load.fei and main.fei. Let us look at them closely one by one along with all its supported commands.
207.2.5.1 Singular Commands

Singular Commands does not require any physical group to operate. All the text following exclamation mark ‘!’ are copied directly to the main.fei (Section 207.2.4.6). For Example:- ‘! include ‘load.fei’;’ is translated as ‘include “load.fei” ’ in main.fei analysis file. See [Example.1.gmessi] for its usage.

**Note:** Real-ESSI DSL/commands must be followed by the exclamation mark ‘!’.
207.2.5.2 Add Node Commands

Add Node Commands have only two commands. \[\text{Add All Node}\{\text{unit,nof dofs}\}\] adds all the nodes generated in mesh (.msh) file to 'node.fei' file. Whereas, \[\text{Add Node}\{\text{Unit,NumDofs}\}\] add all the nodes of only specified physical group by the user. These commands operates on all the nodes of the physical group and generate an equivalent Real-ESSI DSL for each of them.

\textbf{NOTE:-} Every Add Node commands get translated into the \textit{node.fei} (Section 207.2.4.4).

- \textbf{gmESSI} : \[\text{Add Node}\{\text{PhysicalGroup, Unit, NumDofs}\}\]
  translates to series of
  \textbf{Real-ESSI DSL} : add node \# <.> at \(<L>,<L>,<L>\) with \(<.>\) dofs;
  operated over all the nodes defined in the gmsh '.msh' file.

- \textbf{gmESSI} : \[\text{Add All Node}\{\text{Unit, NumDofs}\}\]
  translates to series of
  \textbf{Real-ESSI DSL} : add node \# <.> at \(<L>,<L>,<L>\) with \(<.>\) dofs;
  operated over all the nodes of the defined physical group
207.2.5.3 Nodal Commands: Operates On All Nodes of the defined Physical Group

Nodal commands operates on all the nodes of the physical group defined by the user. For example:

\[ \text{Fix Dofs\{Physical\_Group \#Lateral\_Surface,ux\}} \]

would fix \(ux\) degree of freedom of all the nodes of physical group 'Lateral Surface'. It will generate equivalent Real-ESSI DSL ‘fix node \# < . > dof < . >’ and apply to all the nodes of that physical group. Figure 207.6 shows how gmESSI operated on physical groups.

As earlier stated, that the arguments of gmESSI commands are dummy and gets copied directly to the ESSI equivalent command, so one must be very much aware while writing the arguments to the commands. The arguments should be filled with values of the corresponding ESSI command along with required units if any. For more details about the values to the arguments, please refer to ESSI Manual.

**NOTE:**- Every Nodal command gets translated to the load.fei file (Section 207.2.4.5).

The different commands under this category and their corresponding Real-ESSI commands are listed below

1. **gmESSI** : \[Add\_Nodes\_To\_Physical\_Group\{Physical\_Group , Physical\_Node\_Group\_String\}\]
   translates to series of
   
   **Real-ESSI DSL** : add nodes \(< . . >\) to [physical_node_group] "string";
   
   operated over all the nodes of the defined physical group

2. **gmESSI** : \[Add\_Self\_Weight\_To\_Node\{Physical\_Group , field\#1\}\]
   translates to series of
   
   **Real-ESSI DSL** : add load \# < . > to node \# < . > type [self_weight] use acceleration field \# < . >;
   
   operated over all the nodes of the defined physical group

3. **gmESSI** : \[Add\_Node\_Load\_Linear\{Physical\_Group , Force\_Type , Magnitude\}\]
   translates to series of
   
   **Real-ESSI DSL** : add load \# < . > to node \# < . > type [linear] [FORCETYPE] = < forcemoment >; ///[FORCETYPE] = [Fx] [Fy] [Fz] [Mx] [My] [Mz] [F\_fluid\_x] [F\_fluid\_y] [F\_fluid\_z]
   
   operated over all the nodes of the defined physical group

4. **gmESSI** : \[Add\_Node\_Load\_Path\_Time\_Series\{Physical\_Group , Force\_Type , Magnitude , Series\_File\}\]
   translates to series of
Real-ESSI DSL : add load \( \# < . > \) to node \( \# < . > \) type \([\text{path\_time\_series}] \) \([\text{FORCETYPE}] = < \text{forceormoment} > \) \( \text{series\_file} = "\text{string}"; \)
operated over all the nodes of the defined physical group

5. gmESSI : \([\text{Add\_Node\_Load\_Path\_Series}\{\text{PhysicalGroup}, \text{Force\_Type}, \text{Magnitude}, \text{Time\_Step}, \text{Series\_File}\}]\)
translates to series of
Real-ESSI DSL : add load \( \# < . > \) to node \( \# < . > \) type \([\text{path\_series}] \) \([\text{FORCETYPE}] = < \text{forceormoment} > \) \( \text{time\_step} = < T > \) \( \text{series\_file} = "\text{string}"; \)
operated over all the nodes of the defined physical group

6. gmESSI : \([\text{Add\_Node\_Load\_From\_Reaction}\{\text{PhysicalGroup}\}]\)
translates to series of
Real-ESSI DSL : add load \( \# < . > \) to node \( \# < . > \) type \([\text{from\_reactions}]\);
operated over all the nodes of the defined physical group

7. gmESSI : \([\text{Add\_Node\_Load\_Imposed\_Motion\_Time\_Series}\{\text{PhysicalGroup}, \text{Dof\_Type}, \text{Time\_Step}, \text{Disp\_Scale}, \text{Disp\_File}, \text{Vel\_Scale}, \text{Vel\_File}, \text{Acc\_Scale}, \text{Acc\_File}\}]\)
translates to series of
Real-ESSI DSL : add imposed motion \( \# < . > \) to node \( \# < . > \) dof \( < \text{DOFTYPE} > \) \( \text{time\_step} = < T > \) \( \text{displacement\_scale\_unit} = < L > \) \( \text{displacement\_file} = "\text{string}" \) \( \text{velocity\_scale\_unit} = < L/T > \) \( \text{velocity\_file} = "\text{string}" \) \( \text{acceleration\_scale\_unit} = < L/T^2 > \) \( \text{acceleration\_file} = "\text{string}"; \)
operated over all the nodes of the defined physical group

8. gmESSI : \([\text{Add\_Node\_Load\_Imposed\_Motion\_Time\_Series}\{\text{PhysicalGroup}, \text{Dof\_Type}, \text{Time\_Step}, \text{Disp\_Scale}, \text{Disp\_File}, \text{Vel\_Scale}, \text{Vel\_File}, \text{Acc\_Scale}, \text{Acc\_File}\}]\)
translates to series of
Real-ESSI DSL : add load \( \# < . > \) type imposed motion to node \( \# < . > \) dof \( < \text{DOFTYPE} > \) \( \text{time\_step} = < T > \) \( \text{displacement\_scale\_unit} = < L > \) \( \text{displacement\_file} = "\text{string}" \) \( \text{velocity\_scale\_unit} = < L/T > \) \( \text{velocity\_file} = "\text{string}" \) \( \text{acceleration\_scale\_unit} = < L/T^2 > \) \( \text{acceleration\_file} = "\text{string}"; \)
operated over all the nodes of the defined physical group

9. gmESSI : \([\text{Add\_Node\_Load\_Imposed\_Motion\_Series}\{\text{PhysicalGroup}, \text{Dof\_Type}, \text{Disp\_Scale}, \text{Disp\_File}, \text{Vel\_Scale}, \text{Vel\_File}, \text{Acc\_Scale}, \text{Acc\_File}\}]\)
translates to series of
Real-ESSI DSL : add imposed motion \( \# \) \(<\cdot>\) to node \( \# \) \(<\cdot>\) dof \(<DOFTYPE>\)
\[\text{displacement\_scale\_unit} = \text{<L>}, \text{displacement\_file} = \text{"string"}, \text{velocity\_scale\_unit} = \text{<L/T>}, \text{velocity\_file} = \text{"string"}, \text{acceleration\_scale\_unit} = \text{<L/T^2>}, \text{acceleration\_file} = \text{"string"};\]
operated over all the nodes of the defined physical group

10. **gmESSI** : \([\text{Add\_Node\_Load\_Imposed\_Motion\_Time\_Series}\{\text{PhysicalGroup} , \text{Dof\_Type} , \text{Time\_Step} , \text{Disp\_Scale} , \text{Disp\_File} , \text{Vel\_Scale} , \text{Vel\_File} , \text{Acc\_Scl} , \text{Acc\_File}\}]\)
translates to series of
Real-ESSI DSL : add load \(\# \) \(<\cdot>\) type imposed motion to node \(\# \) \(<\cdot>\) dof \(<DOFTYPE>\)
\[\text{displacement\_scale\_unit} = \text{<L>}, \text{displacement\_file} = \text{"string"}, \text{velocity\_scale\_unit} = \text{<L/T>}, \text{velocity\_file} = \text{"string"}, \text{acceleration\_scale\_unit} = \text{<L/T^2>}, \text{acceleration\_file} = \text{"string"};\]
operated over all the nodes of the defined physical group

11. **gmESSI** : \([\text{Add\_Damping\_To\_Node}\{\text{PhysicalGroup} , \text{damping}\#1}\}]\)
translates to series of
Real-ESSI DSL : add damping \(\# \) \(<\cdot>\) to node \(\# \) \(<\cdot>\);
operated over all the nodes of the defined physical group

12. **gmESSI** : \([\text{Add\_Mass\_To\_Node}\{\text{PhysicalGroup} , \text{MassX} , \text{MassY} , \text{MassZ}\}]\)
translates to series of
Real-ESSI DSL : add mass to node \(\# \) \(<\cdot>\) \(mx = \text{<M>}, my = \text{<M>}, mz = \text{<M>}\);
operated over all the nodes of the defined physical group

13. **gmESSI** : \([\text{Add\_Beam\_Mass\_To\_Node}\{\text{PhysicalGroup} , \text{MassX} , \text{MassY} , \text{MassZ} , \text{ImassX} , \text{ImassY} , \text{ImassZ}\}]\)
translates to series of
Real-ESSI DSL : add mass to node \(\# \) \(<\cdot>\) \(mx = \text{<M>}, my = \text{<M>}, mz = \text{<M>}, \text{Imx} = \text{<ML^2>}, \text{Imy} = \text{<ML^2>}, \text{Imz} = \text{<ML^2>}\);
operated over all the nodes of the defined physical group

14. **gmESSI** : \([\text{Fix\_Dofs}\{\text{PhysicalGroup} , \text{Dof\_Types}\}]\)
translates to series of
Real-ESSI DSL : fix node \(\# \) \(<\cdot>\) dofs \(<Dof\_Types>\);
operated over all the nodes of the defined physical group

15. **gmESSI** : \([\text{Free\_Dofs}\{\text{PhysicalGroup} , \text{Dof\_Types}\}]\)
translates to series of
**Real-ESSI DSL**: 
free node \# <.> dofs <.>;
operated over all the nodes of the defined physical group

16. **gmESSI**: 
[Remove Node{PhysicalGroup}]
translates to series of
**Real-ESSI DSL**: remove node \# <.>;
operated over all the nodes of the defined physical group

17. **gmESSI**: 
[Remove Equal Dof Constrain{PhysicalGroup}]
translates to series of
**Real-ESSI DSL**: remove constraint [equal_dof] node \# <.>;
operated over all the nodes of the defined physical group

18. **gmESSI**: 
[Remove Displacement From Node{PhysicalGroup}]
translates to series of
**Real-ESSI DSL**: remove displacement from node \# <.>;
operated over all the nodes of the defined physical group
207.2.5.4 General Elemental Commands: Operates On All Elements of the defined Physical Group

General Elemental Commands operates on all the elements of a physical group. The translations are written in `load.fei` file. For example:- [Add_SelfWeight_To_Element{Physical_Group#Soil,Field:= 1}] would add self-weight to all the elements of the physical group ‘Soil’ along the field#1 direction using series of equivalent Real-ESSI DSL ‘add load # < . > to element # < . > type [self_weight] use acceleration field # < . >;’

The different commands under this category and their corresponding ESSI commands are listed below:

1. `gmESSI` : [Add_Elements_To_Physical_Group{PhysicalGroup , Physical_Element_Group_String}] translates to series of
   
   **Real-ESSI DSL** : add elements (<.>) to [physical_element_group] "string";
   
   operated over all the nodes of the defined physical group

2. `gmESSI` : [Add_Self_Weight_To_Element{PhysicalGroup , field#1}] translates to series of
   
   **Real-ESSI DSL** : add load # < . > to element # < . > type [self_weight] use acceleration field # < . >;
   
   operated over all the nodes of the defined physical group

3. `gmESSI` : [Add_Damping_To_Element{PhysicalGroup , damping#1}] translates to series of
   
   **Real-ESSI DSL** : add damping # < . > to element # < . >;
   
   operated over all the nodes of the defined physical group

4. `gmESSI` : [Remove_Element{PhysicalGroup}] translates to series of
   
   **Real-ESSI DSL** : remove element # < . >;
   
   operated over all the nodes of the defined physical group

5. `gmESSI` : [Remove_Strain_From_Element{PhysicalGroup}] translates to series of
   
   **Real-ESSI DSL** : remove strain from element # < . >;
   
   operated over all the nodes of the defined physical group
207.2.5.5 Elemental Commands: Operates On All Elements of the defined Physical Group

Elemental Commands operates only to specific elements of a physical group. The translations are written in `element.feI` file. For example:- `[Add_8NodeBrick{Physical_Group#Soil,1}]` would initialize all the hexahedron elements of physical group `Soil` to equivalent Real-ESSI commands for defining 8-noded bricks elements `add element # < . > type [8NodeBrick] with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >)` use material `# < . >`. Figure 207.6 shows how gmESSI operated on physical groups. The different commands under this category and their corresponding ESSI commands are listed below

1. **gmESSI**: `[Add_20NodeBrick{PhysicalGroup, Num_Gauss_Points, material#1}]`
   
   translates to series of
   
   **Real-ESSI DSL**: `add element # < . > type [20NodeBrick] with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >)` use material `# < . >;`
   
   operated over all the elements of the defined physical group

2. **gmESSI**: `[Add_20NodeBrick_Variable_GaussPoints{PhysicalGroup, Num_Gauss_Points, material#1}]`
   
   translates to series of
   
   **Real-ESSI DSL**: `add element # < . > type [20NodeBrick] using < . > Gauss points each direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >)` use material `# < . >;`
   
   operated over all the elements of the defined physical group

3. **gmESSI**: `[Add_20NodeBrick_upU{PhysicalGroup, material#1, Porosity, Alpha, Solid_Density, Fluid_Density, Perm_X, Perm_Y, Perm_Z, Solid_Bulk_Modulus, Fluid_Bulk_Modulus}]`
   
   translates to series of
   
   **Real-ESSI DSL**: `add element # < . > type [20NodeBrick_upU] with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >)` use material `# < . > and porosity = < . > alpha = < . > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x = < L^3T/M > k_y = < L^3T/M > k_z = < L^3T/M > K_s = < stress > K_f = < stress >;`
   
   operated over all the elements of the defined physical group

   
   translates to series of
   
   **Real-ESSI DSL**: `add element # < . > type [20NodeBrick_upU] with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >)` use material `# < . > and porosity = < . > alpha = < . > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x = < L^3T/M > k_y = < L^3T/M > k_z = < L^3T/M > K_s = < stress > K_f = < stress >;`
   
   operated over all the elements of the defined physical group
Solid_Bulk_Modulus, Fluid_Bulk_Modulus]

translates to series of

**Real-ESSI DSL** : add element # <.> type [20NodeBrick_upU] using <.> Gauss points each
operated over all the elements of the defined physical group

8. gmESSI : [Add_27NodeBrick_upU{PhysicalGroup, material#1, Porosity, Alpha, Solid_Density, Fluid_Density, Perm_X, Perm_Y, Perm_Z, Solid_Bulk_Modulus, Fluid_Bulk_Modulus}] translates to series of

Real-ESSI DSL : add element # < > type [27NodeBrick_upU] with nodes (< >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >) use material # < > and porosity = < > alpha = < > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x = < L^3 T/M > k_y = < L^3 T/M > k_z = < L^3 T/M > K_s = < stress > K_f = < stress >;
operated over all the elements of the defined physical group

9. gmESSI : [Add_27NodeBrick_upU_Variable_GaussPoints{PhysicalGroup, NumGaussPoints, material#1, Porosity, Alpha, Solid_Density, Fluid_Density, Perm_X, Perm_Y, Perm_Z, Solid_Bulk_Modulus, Fluid_Bulk_Modulus}] translates to series of

Real-ESSI DSL : add element # < > type [27NodeBrick_upU] using < > Gauss points each direction with nodes (< >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >) use material # < > and porosity = < > alpha = < > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x = < L^3 T/M > k_y = < L^3 T/M > k_z = < L^3 T/M > K_s = < stress > K_f = < stress >;
operated over all the elements of the defined physical group

10. gmESSI : [Add_27NodeBrick_up{PhysicalGroup, material#1, Porosity, Alpha, Solid_Density, Fluid_Density, Perm_X, Perm_Y, Perm_Z, Solid_Bulk_Modulus, Fluid_Bulk_Modulus}] translates to series of

Real-ESSI DSL : add element # < > type [27NodeBrick_up] with nodes (< >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >, < >) use material # < > and porosity = < > alpha = < > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x = < L^3 T/M > k_y = < L^3 T/M > k_z = < L^3 T/M > K_s = < stress > K_f = < stress >;
operated over all the elements of the defined physical group

translates to series of

**Real-ESSI DSL**: add element # < . > type [27NodeBrick_up] using < . > Gauss points each direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . > and porosity = < . > alpha = < . > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x = < L^3T/M > k_y = < L^3T/M > k_z = < L^3T/M > K_s = < stress > K_f = < stress >;

operated over all the elements of the defined physical group

12. **gmESSI**: \[Add\_Equal\_Dof\{PhysicalGroup , Dof\_Type\}\]
translates to series of

**Real-ESSI DSL**: add constraint [equal_dof] with master node # < . > and slave node # < . > dof to constrain < . >;

operated over all the elements of the defined physical group

13. **gmESSI**: \[Add\_Equal\_Dof\{PhysicalGroup , Master\_Dof , \}\]
translates to series of

**Real-ESSI DSL**: add constraint [equal_dof] with node # < . > dof < . > master and node # < . > dof < . > slave;

operated over all the elements of the defined physical group

14. **gmESSI**: \[Add\_ShearBeam\{PhysicalGroup , CrossSection , material#1\}\]
translates to series of

**Real-ESSI DSL**: add element # < . > type [ShearBeam] with nodes (< . >, < . >) cross_section = < l^2 > use material # < . >;

operated over all the elements of the defined physical group

15. **gmESSI**: \[Add\_DispBeamColumn3D\{PhysicalGroup , Num\_Integr\_Points , Section\_Number , Density , XZ\_Plane\_Vect\_x , XZ\_Plane\_Vect\_y , XZ\_Plane\_Vect\_z , Joint1\_Offset\_x , Joint1\_Offset\_y , J1\_z , Joint2\_Offset\_x , J2\_y , J2\_Offset\_z\}\]
translates to series of

**Real-ESSI DSL**: add element # < . > type [BeamColumnDispFiber3d] with nodes (< . >, < . >) number_of_integration_points = < . > section_number = < . > mass_density = < M/L^3 > xz_plane_vector = (< . >, < . >, < . >) joint_1_offset = (< L >, < L >, < L >) joint_2_offset = (< L >, < L >, < L >);
operated over all the elements of the defined physical group

16. **gmESSI**: 
\[
\text{Add Beam Elastic}\{\text{PhysicalGroup, Cross Section, Elastic Modulus, Shear Modulus, Jx, ly, lz, Density, XZ Plane Vect.x, XZ Plane Vect.y, XZ Plane Vect.z, Joint1 Offset.x, Joint1 Offset.y, Joint1 Offset.z, Joint2 Offset.x, Joint2 Offset.y, J2 Offset.z}\}
\]
translates to series of

Real-ESSI DSL: add element \# \(<\ .\ >\) type [beam elastic] with nodes \(<\ .\ >, <\ .\ >\)

cross section = \(<\ area\ >\) elastic_modulus = \(<\ F/L^2\ >\) shear_modulus = \(<\ F/L^2\ >\) torsion Jx = \(<\ length^4\ >\) bending ly = \(<\ length^4\ >\) bending lz = \(<\ length^4\ >\) mass density = \(<\ M/L^3\ >\)
xz plane vector = \(<\ .\ >, <\ .\ >, <\ .\ >\) joint 1 offset = \(<\ L\ >, <\ L\ >, <\ L\ >\) joint 2 offset = \(<\ L\ >, <\ L\ >, <\ L\ >\);
operated over all the elements of the defined physical group

17. **gmESSI**: 
\[
\text{Add Beam Elastic Lumped Mass}\{\text{PhysicalGroup, Cross Section, Elastic Modulus, Shear Modulus, Jx, ly, lz, Density, XZ Plane Vect.x, XZ Plane Vect.y, XZ Plane Vect.z, Joint1 Offset.x, Joint1 Offset.y, Joint1 Offset.z, Joint2 Offset.x, Joint2 Offset.y, J2 Offset.z}\}
\]
translates to series of

Real-ESSI DSL: add element \# \(<\ .\ >\) type [beam elastic lumped mass] with nodes \(<\ .\ >, <\ .\ >\)

cross section = \(<\ area\ >\) elastic_modulus = \(<\ F/L^2\ >\) shear_modulus = \(<\ F/L^2\ >\) torsion Jx = \(<\ length^4\ >\) bending ly = \(<\ length^4\ >\) bending lz = \(<\ length^4\ >\) mass density = \(<\ M/L^3\ >\)
xz plane vector = \(<\ .\ >, <\ .\ >, <\ .\ >\) joint 1 offset = \(<\ L\ >, <\ L\ >, <\ L\ >\) joint 2 offset = \(<\ L\ >, <\ L\ >, <\ L\ >\);
operated over all the elements of the defined physical group

18. **gmESSI**: 
\[
\text{Add Beam Displacement Based}\{\text{PhysicalGroup, Num Integration Points, Section Number, Density}\}
\]
translates to series of

Real-ESSI DSL: add element \# \(<\ .\ >\) type [beam displacement based] with nodes \(<\ .\ >, <\ .\ >\)

with \# \(<\ .\ >\) integration points use section \# \(<\ .\ >\) mass density = \(<\ M/L^3\ >\) IntegrationRule = "" xz plane vector = \(<\ .\ >, <\ .\ >, <\ .\ >\) joint 1 offset = \(<\ L\ >, <\ L\ >, <\ L\ >\) joint 2 offset = \(<\ L\ >, <\ L\ >, <\ L\ >\);
operated over all the elements of the defined physical group

19. **gmESSI**: 
\[
\text{Add Hard Contact}\{\text{PhysicalGroup, Normal Stiffness, Tangential Stiffness, Normal Damping, Tangential Damping, Friction Ratio, Norm Vec.x, Norm Vec.y, Norm Vec.z}\}
\]
translates to series of
**Real-ESSI DSL**: add element \# < . > type [HardContact] with nodes (< . >, < . >) normal_stiffness = \(< F/L >\) tangential_stiffness = \(< F/L >\) normal_damping = \(< F/L >\) tangential_damping = \(< F/L >\) friction_ratio = < . > contactPlane_vector = (< . >, < . >, < . >);
operated over all the elements of the defined physical group

20. **gmESSI**: [Add_CoupledHardContact{PhysicalGroup, Normal_Stiffness, Tangential_Stiffness, Normal_Damping, Tangential_Damping, Friction_Ratio, Norm_Vect_x, Norm_Vect_y, Norm_Vect_z}]
translates to series of
**Real-ESSI DSL**: add element \# < . > type [CoupledHardContact] with nodes (< . >, < . >)
normal_stiffness = \(< F/L >\) normal_penalty_stiffness = \(< F/L >\) tangential_stiffness = \(< F/L >\) normal_damping = \(< F/L >\) tangential_damping = \(< F/L >\) friction_ratio = < . > contactPlane_vector = (< . >, < . >, < . >);
operated over all the elements of the defined physical group

21. **gmESSI**: [Add_SoftContact{PhysicalGroup, Initial_Normal_Stiffness, Stiffning_Rate, Maximum_Normal_Stiffness, Tangential_Stiffness, Normal_Damping, Tangential_Damping, Friction_Ratio, Norm_Vect_x, Norm_Vect_y, Norm_Vect_z}]
translates to series of
**Real-ESSI DSL**: add element \# < . > type [SoftContact] with nodes (< . >, < . >) initial_normal_stiffness = \(< F/L >\) stiffening_rate = \(< 1/L >\) max_normal_stiffness = \(< F/L >\) tangential_stiffness = \(< F/L >\) normal_damping = \(< F/L >\) tangential_damping = \(< F/L >\) friction_ratio = < . > contactPlane_vector = (< . >, < . >, < . >);
operated over all the elements of the defined physical group

22. **gmESSI**: [Add_CoupledSoftContact{PhysicalGroup, Initial_Normal_Stiffness, Stiffning_rate, Maximum_Normal_Stiffness, Tangential_Stiffness, Normal_Damping, Tangential_Damping, Friction_Ratio, Norm_Vect_x, Norm_Vect_y, Norm_Vect_z}]
translates to series of
**Real-ESSI DSL**: add element \# < . > type [CoupledSoftContact] with nodes (< . >, < . >) initial_normal_stiffness = \(< F/L >\) stiffening_rate = \(< 1/L >\) max_normal_stiffness = \(< F/L >\) tangential_stiffness = \(< F/L >\) normal_damping = \(< F/L >\) tangential_damping = \(< F/L >\) friction_ratio = < . > contactPlane_vector = (< . >, < . >, < . >);
operated over all the elements of the defined physical group

23. **gmESSI**: [Add_Truss{PhysicalGroup, material#1, Cross_Sectin, Density}]
translates to series of

**Real-ESSI DSL**: add element # <.> type [truss] with nodes (<.>, <.>) use material # <.>
cross_section = <length^2> mass_density = <M/L^3>;
operated over all the elements of the defined physical group

24. **gmESSI**: [Add_8NodeBrick{PhysicalGroup , material#1}]

translates to series of

**Real-ESSI DSL**: add element # <.> type [8NodeBrick] with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>;
operated over all the elements of the defined physical group

25. **gmESSI**: [Add_Cosserat8NodeBrick{PhysicalGroup , material#1}]

translates to series of

**Real-ESSI DSL**: add element # <.> type [Cosserat8NodeBrick] with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>;
operated over all the elements of the defined physical group

26. **gmESSI**: [Add_8NodeBrick_Variable_GaussPoints{PhysicalGroup , NumGaussPoints , material#1}]

translates to series of

**Real-ESSI DSL**: add element # <.> type [8NodeBrick] using <.> Gauss points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>;
operated over all the elements of the defined physical group

27. **gmESSI**: [Add_8NodeBrick_upU{PhysicalGroup , material#1 , Porosity , Alpha , Solid_Density , Fluid_Density , Perm_X , Perm_Y , Perm_Z , Solid_Bulk_Modulus , Fluid_Bulk_Modulus}]

translates to series of

**Real-ESSI DSL**: add element # <.> type [8NodeBrick_upU] with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.> porosity = <.> alpha = <.> rho_s = <M/L^3> rho_f = <M/L^3> k_x = <L^3T/M> k_y = <L^3T/M> k_z = <L^3T/M> K_s = <stress> K_f = <stress>;
operated over all the elements of the defined physical group


translates to series of

Real-ESSI DSL : add element # < . > type [8NodeBrick_upU] using < . > Gauss points each
direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material
# < . > porosity = < . > alpha = < . > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x =
< L^3T/M > k_y = < L^3T/M > k_z = < L^3T/M > K_s = < stress > K_f = < stress >;
operated over all the elements of the defined physical group

29. gmESSI : [Add_8NodeBrick_up{PhysicalGroup, material#1, Porosity, Alpha, Solid_Density,
Fluid_Density, Perm_X, Perm_Y, Perm_Z, Solid_Bulk_Modulus, Fluid_Bulk_Modulus}]
translates to series of

Real-ESSI DSL : add element # < . > type [8NodeBrick_up] with nodes (< . >, < . >, < . >,
< . >, < . >, < . >, < . >, < . >) use material # < . > porosity = < . > alpha = < . > rho_s
= < M/L^3 > rho_f = < M/L^3 > k_x = < L^3T/M > k_y = < L^3T/M > k_z = < L^3T/M >
K_s = < stress > K_f = < stress >;
operated over all the elements of the defined physical group

30. gmESSI : [Add_8NodeBrick_up_Variable_GaussPoints{PhysicalGroup, Num_Gauss_Points, mate-
rial#1, Porosity, Alpha, Solid_Density, Fluid_Density, Perm_X, Perm_Y, Perm_Z, Solid_Bulk_Modulus
, Fluid_Bulk_Modulus}]
translates to series of

Real-ESSI DSL : add element # < . > type [8NodeBrick_up] using < . > Gauss points each
direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material
# < . > porosity = < . > alpha = < . > rho_s = < M/L^3 > rho_f = < M/L^3 > k_x =
< L^3T/M > k_y = < L^3T/M > k_z = < L^3T/M > K_s = < stress > K_f = < stress >;
operated over all the elements of the defined physical group
207.2.5.6 Elemental Compound Commands: Operates On All Surface Elements of the defined Physical Group [Surface Loads]

Elemental Compound Commands operates on two physical groups, one for surface and another for the element on which surface is present. It is used mainly for adding surface loads, which require surface number as well as element no in Real-ESSI DSL. For example: 

```
[Add_8NodeBrick_SurfaceLoad{Physical_Group#Volume, Physical_Group#Surface, 10*Pa}]
```

would initialize surface load of 10Pa on surfaces defined by physical_group ‘Surface’ on elements defined by physical_group ‘Volume’.

The different commands under this category and their corresponding ESSI commands are listed below

**NOTE:** Every Elemental commands get translated into the load.fei (Section 207.2.4.5).

1. `gmESSI`: `[Add_20NodeBrick_SurfaceLoad{PhysicalGroup#Volume, PhysicalGroup#Surface, Pressure}]`
   translates to series of
   
   **Real-ESSI DSL** : add load # < . > to element # < . > type [surface] at nodes (< . >, < . > , < . >, < . >, < . >, < . >, < . >) with magnitude < Pa >;
   operated over all the elements of the defined physical group

2. `gmESSI`: `[Add_20NodeBrick_SurfaceLoad{PhysicalGroup#Volume, PhysicalGroup#Surface, Press1 , Press2 , Press3 , Press4 , Press5 , Press6 , Press7 , Press8}]`
   translates to series of
   
   **Real-ESSI DSL** : add load # < . > to element # < . > type [surface] at nodes (< . >, < . > , < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) with magnitudes (< Pa >, < Pa >, < Pa >, < Pa >, < Pa >, < Pa >, < Pa >, < Pa >, < Pa >);
   operated over all the elements of the defined physical group

3. `gmESSI`: `[Add_27NodeBrick_SurfaceLoad{PhysicalGroup#Volume, PhysicalGroup#Surface, Pressure}]`
   translates to series of
   
   **Real-ESSI DSL** : add load # < . > to element # < . > type [surface] at nodes (< . >, < . > , < . >, < . >, < . >, < . >, < . >, < . >, < . >) with magnitude < Pa >;
   operated over all the elements of the defined physical group

   translates to series of
**Real-ESSI DSL**: add load # <.> to element # <.> type [surface] at nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) with magnitudes (<Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>, <Pa>); operated over all the elements of the defined physical group

5. **gmESSI**: [Add 8NodeBrick_SurfaceLoad{PhysicalGroup#Volume, PhysicalGroup#Surface, Pressure}]
   translates to series of
   **Real-ESSI DSL**: add load # <.> to element # <.> type [surface] at nodes (<.>, <.>, <.>, <.>, <.>) with magnitude <Pa>;
   operated over all the elements of the defined physical group

6. **gmESSI**: [Add 8NodeBrick_SurfaceLoad{PhysicalGroup#Volume, PhysicalGroup#Surface, Press1, Press2, Press3, Press4}]
   translates to series of
   **Real-ESSI DSL**: add load # <.> to element # <.> type [surface] at nodes (<.>, <.>, <.>, <.>) with magnitudes (<Pa>, <Pa>, <Pa>, <Pa>);
   operated over all the elements of the defined physical group
207.2.5.7 Special Commands

The translator supports some special commands to perform some special functions that are regularly required in simulations. It supports the Connect Command (Section 207.2.5.8) allows to join or create nodes between two physical groups.
207.2.5.8 Connect Command

Connect Commands creates/find layers of 2-noded elements between any two parallel geometrical physical entities like two lines, two surface or two volumes and creates a physical group of those elements and updates this information in the XYZ.msh file. Since gmsh does not include the feature of defining or creating 2-noded elements after the mesh creation, this command can be very useful in that case. For example:- defining contacts/interfaces, embedded piles, boundary conditions, connections etc. The command syntax for connect command is

gmESSI :: [Connect{Physical_Group#tag_From, Physical_Group#tag_To, Physical_Group#tag_Between, dir_vect, mag, no_times, algo{find|create}, tolerance,New_Physical_Group_Name}]

- **Physical_Group#tag_From** :: It defines the starting nodes
- **Physical_Group#tag_To** :: It defines the set of end nodes
- **Physical_Group#tag_Between** :: It defines the set of nodes where the intermediary nodes can be found, while searching. While creating nodes, it does not play any role.
- **dir_vect** :: It defines the direction in which the user wants to create or find the nodes. The direction vector argument is given as \{x\_comp \ y\_comp \ z\_comp\}. Example:- \{0 \ 0 \ -1\}, \{1 \ 1 \ 0\} .. etc.
- **mag** :: It defines the length of each 2-noded line elements
- **no_times** :: It defines number of layers of 2-noded elements, the user want to create/find
- **algo{find|create}** :: It defines the algo which is either 0 or 1 meaning whether to find or create the intermediary node
- **tolerance** :: It defines the tolerance is required to finding the nodes. It should be less than the minimum of the distance of neighboring nodes.
- **New_Physical_Group_Name** :: This argument enables the user to give a name to the 2-noded new-physical group formed

Figure 207.7 graphically describes arguments of connect command.

This command updates and creates additional nodes and 2-noded elements and also assigns a physical group name "$New_Physical_Group_Name$". gmESSI automatically adds the next id available to the new physical group. The user can then manipulate this newly created physical group with any other gmESSI commands.
The working of this command would be more clear through examples. [Example_3] can be downloaded here. [Example_4] can be downloaded here. These examples describes two situation one where new nodes are to be created and the other where already present nodes needs to be found respectively. In both the cases 2-noded line elements are always created. The examples can also be alternatively located in Examples folder of gmESSI directory.

[Example_3] is a simple example where a tower of certain height above ground surface and also its base embedded in soil is modeled. It starts with a mesh file that creates a node for tower at a certain height and then using algorithm `-create` new nodes are created at certain intervals to generate the beam elements. On the other hand, the embedded beam is created by `-find` algorithm. Let us look at the [Example_3.geo] file.

```bash
$ cat Example_3.geo

// Size of the soil block in meter
Size = 10;

// Height of the Tower in meter
Height = 6;

// Mesh Size of the soil block
Mesh_Size = 1;

// Adding Points and extruding
Point(1)=(-Size/2,-Size/2,-Size/2);
```
Extrude{Size,0,0}{Point{1};Layers{Size/Mesh_Size};Recombine;}
Extrude{0,Size,0}{Line{1};Layers{Size/Mesh_Size};Recombine;}
Extrude{0,0,Size}{Surface{5};Layers{Size/Mesh_Size};Recombine;}

// Make the tower located at height 6 m from the ground surface
Tower = newp;
Point (Tower) = {0,0,Size/2+Height};

/// Create Physical Groups
Physical Volume ("Soil") = {1};
Physical Surface ("Soil_Base_Surface") = {5};
Physical Surface ("Soil_Top_Surface") = {27};
Physical Point ("Tower") = {Tower};

Running [Example_3.gmessi] with the .msh output of geometry file would produce additional nodes and elements as shown in Figure 207.8. An excerpt showing use of connect command with create algo in [Example_3.gmessi] is shown below. The effect of the command is shown in Figure 207.8.

$ cat Example_3.gmessi
..........
[Connect{Physical_Group#Tower, Physical_Group#Soil_Top_Surface, Physical_Group#Soil_Top_Surface, dv1:= 0 \ 0 \ -1, mag:= 2, Tolerance:= 0, algo:= create, noT:= 3, PhysicalGroupName:= Tower_Beam_Above_Ground}]
..............

The terminal displays the information about number of elements and nodes created and also displays the information about the new physical group information i.e. id and name. The new physical group creation can be seen in the [Example_3.gmsh] in Example_3_ESSI_Simulation folder. The terminal message and mesh file is shown below. It also displays error message if more than one node is found in the tolerance provided.

$ gmessy Example_3.gmessi
New Physical Group "Tower_Beam_Above_Ground" having id 5 consisting of 4 Nodes and 3 2-noded elements created

$ cat Example_3_ESSI_Simulation\Example_3.msh
..........
$PhysicalNames
7
0 4 "Tower"
2 2 "Soil_Base_Surface"
7 3 "Soil_Top_Surface"
3 1 "Soil"
1 5 "Tower_Beam_Above_Ground"
3 6 "Tower_Base_Node"
1 7 "Tower_Embedded_Beam"
$EndPhysicalNames
(a) Initial mesh file Example.3.msh generated by gmsh. The dir vector is in Z axis 0,0,-1

(b) Final mesh after gmESSI

Figure 207.8: Example 3 Contact Problem. (b) shows the nodes and elements generated by gmESSI Translator.

[Example 4] describes a foundation on soil problem with contact/interface between them. The contact element is created with the help of comment command using algo "-find". Let us look at the [Example 4.geo] file.

```geo
$ cat Example_4.geo

// Size of the soil block
Size = 1;

// Thickness of Foundation
Thick = 0.1;
Foundation_Layers = 2;

/// Mesh Size of the block
Mesh_Size = 0.2;

// Adding Points and extruding
Point(1)={-Size/2,-Size/2,-Size/2};
Extrude{Size,0,0}{Point{1};Layers{Size/Mesh_Size};Recombine;}
```
Extrude{0,Size,0}{Line{1};Layers{Size/Mesh_Size};Recombine;}
Extrude{0,0,Size}{Surface{5};Layers{Size/Mesh_Size};Recombine;}

// Make sure in Tools -> Geometry -> General
// Geometry tolerance is set smaller than Epsilon
// such as Geometry tolerance = 1e-14

Epsilon = 1e-8;
Translate {0, 0, Epsilon} {Duplicata{Surface{27};}}
Transfinite Line {29,30,31,32} = Size/Mesh_Size +1;
Transfinite Surface {28};
Recombine Surface {28};

///// Extruding the surface to foundation thickness
Extrude{0,0,Thick}{Surface{28};Layers{Foundation_Layers};Recombine;}

///// Create Physical Groups
Physical Volume ("Soil") = {1};
Physical Surface ("Soil_Base_Surface") ={5};
Physical Surface ("Soil_Top_Surface") ={27};
Physical Surface ("Foundation_Base_Surface")={28};
Physical Surface ("Foundation_Top_Surface") ={54};
Physical Volume ("Foundation") = {2};

Physical Surface("Fix_X") = {26, 53, 45, 18};
Physical Surface("Fix_Y") = {22, 49, 14, 41};
Physical Volume("3_Dofs") = {1,2};

The above geometry file is then meshed with gmsh to get the .msh file. In this file, the connect command is applied between physical group Foundation_Base_Surface and Soil_Top_Surface to create contact/interface elements. The corresponding connect command would be as

```
$ gmessy Example_4.gmessi
..........
[Connect{Physical_Group#Soil_Top_Surface, ←
Physical_Group#Foundation_Base_Surface, ←
Physical_Group#Foundation_Base_Surface, dv1:= 0\0\1, mag:= 0, Tolerance:= ←
0.001, algo:= find, noT:= 1, PhysicalGroupName:= Contact_Elements}]
.............
```

Similarly the updated [Example_4.msh] contains the new physical group and terminal shows the new physical group of 2-noded elements created. Figure 207.9 shows the new nodes found and creation of 2-noded elements.

```
$ gmessy Example_3.gmessi
New Physical Group "Contact_Elements" having id 10 consisting of 72 Nodes and ←
36 2-noded elements created
```
Figure 207.9: Example 4 finding nodes problem. (b) shows the nodes and elements generated by gmESSI Translator.

1 $ cat Example_4_ESSI_Simulation\Example_4.msh
2 10
3 2 2 "Soil_Base_Surface"
4 2 3 "Soil_Top_Surface"
5 2 4 "Foundation_Base_Surface"
6 2 5 "Foundation_Top_Surface"
7 2 7 "Fix_X"
8 2 8 "Fix_Y"
9 3 1 "Soil"
10 3 6 "Foundation"
11 3 9 "3_Dofs"
12 1 10 "Contact_Elements"
13 ............

NOTE: Since the algo is to only find the nodes, so no new nodes are created, but only elements are created. The same message can be seen on the terminal.

207.2.5.9 Write Command

Write command takes filename as an argument and writes the content of a physical group in two separate files one containing all the nodes info and other containing all the elements info and places in the same XYZ_ESSI_Simulation folder. The command syntax is
gmESSI:: [Write_Data{PhyEntyTag,filename}]

- Creates files $XYZ\_filename\_Nodes.txt$ and $XYZ\_filename\_Elements.txt$

- $XYZ\_filename\_Nodes.txt$ :: Contains data for all nodes in a physical group. Each node data is represented in one line as

  \[Node\_no\ x\_coord\ y\_coord\ z\_coord\]

  with meanings as usual.

- $XYZ\_filename\_Elements.txt$ :: Contains data for all elements in a physical group. Each element data is represented in one line as

  \[Element\_no\ Element\_type\ node1\ node2\ node3\ ..\]

  with meanings as usual. $Element\_type$ refers to the same as in Gmsh Manual.

[Example\_4.gmessi] shows the usage of write command.

### 207.2.5.10 Write DRM HDF5 Command

Domain reduction method (DRM) is a very useful method to input 3D seismic excitations into earthquake soil structure interacting system. With a defined physical group as DRM layer, a HDF5 file containing geometric information of the DRM layer, can be generated with the following commands for 1D, 2D and 3D mesh, respectively:

- gmESSI::[Generate\_DRM\_HDF5\_1D{Physical\_Group#<PhyEnty Name or Tag>, Surface\_Normal:=<X | Y | Z>, Node\_Coordinate\_Tol:=<tolerance>, FileName:=<HDF5 file name>}]  

- gmESSI::[Generate\_DRM\_HDF5\_2D{Physical\_Group#<PhyEnty Name or Tag>, Surface\_Plane:=<XY|XZ|YZ>, Surface\_Normal:=<X | Y | Z>, Node\_Coordinate\_Tol:=<tolerance>, FileName:=<HDF5 file name>}]  

- gmESSI::[Generate\_DRM\_HDF5\_3D{Physical\_Group#<PhyEnty Name or Tag>, Surface\_Normal:=<X | Y | Z>, Node\_Coordinate\_Tol:=<tolerance>, FileName:=<HDF5 file name>}]  

Where:

- **Physical\_Group#** defines the physical group name or tag for the DRM layer.

- **Surface\_Normal:=** defines the surface normal direction of the DRM layer. It can be X or Y or Z.

- **Surface\_Plane:=** defines the surface plane of the 2D DRM layer. It can be XY or YZ or XZ.
• **Node\_Coordinate\_Tol:** defines the tolerance to distinguish two different DRM nodes. The tolerance should be much smaller than the FEM mesh size!

• **FileName:** defines the file name of the HDF5 file to be generated.
207.2.6 Steps For Using gmESSI tool

Using gmESSI it is very easy to convert a .msh file to ESSI (.fei) file. This section guides the user through a simple [Example_1.geo], to show the steps necessary for generating Real-ESSI files directly from .msh file through gmESSI. Lets define a problem as shown in Figure 207.10. The [Example_1.geo] can be located in the gmESSI ‘Examples’ directory. Alternatively, it can be downloaded here.

![Figure 207.10: Example_1 description of a block of soil with surface load.](image)

It is a block of dimension \(10m \times 10m \times 10m\) of soil mass whose all 4 lateral faces are fixed in \(ux, uy\) dofs. The bottom face is fixed in \(ux, uy, uz\) dofs. A uniform pressure surface load of \(10Pa\) is applied. The density and elastic modulus of the soil increases from \(2000 \times kg/m^3\) and Young’s modulus is taken as \(200MPa\) as shown in Figure 207.10.

207.2.6.1 Building geometry (.geo) file in Gmsh

The first step is to make the geometry file in Gmsh. While creating the geometry the user should also define all the physical groups on which they intend to either apply boundary condition, define elements, loads etc. In [Example_1.geo], 3 physical groups are needed : one for applying surface load, one for fixities, and one for defining the soil volume and assigning material. The content of [Example_1.geo] file is shown below

```
$cat Example_1.geo
// Size of the block
Size = 10;

/// Mesh Size of the block
Mesh_Size = 2;
```
8
9  // Adding Points and extruding
10  Point(1)={-Size/2,-Size/2,-Size/2};
11  Extrude{Size,0,0}{Point{1};Layers{Size/Mesh_Size};Recombine;}
12  Extrude{0,Size,0}{Line{1};Layers{Size/Mesh_Size};Recombine;}
13  Extrude{0,0,Size}{Surface{5};Layers{Size/Mesh_Size};Recombine;}
14
15  /// Create Physical Groups
16  Physical Volume ("Soil") = {1};
17  Physical Surface ("Base_Surface") = {5};
18  Physical Surface ("Lateral_Surface") = {18,22,14,26};
19  Physical Surface ("Top_Surface") ={27};

207.2.6.2 Generate mesh (.msh) file in Gmsh

Once .geo file is ready with all the physical groups, next step is to mesh the model. The mesh operation will generate the mesh file (.msh) that contains all the mesh information.

The model can be meshed from the terminal directly by running:

gmsh Example_1.geo -3

Here -3 means we are meshing a 3D object, which will automatically mesh all the 3D volumes, 2D surfaces and 1D lines object defined in the geometry model. If there are only 2D surfaces and/or 1D lines object defined in the geometry (.geo) file, use -2 instead. If there are only 1D lines object defined in the geometry (.geo) file, use -1 instead.

A quick look at the generated [Example_1.msh] file containing physical groups is shown below:

$cat Example_5.geo
..............
$PhysicalNames
4
2 2 "Base_Surface"
2 3 "Lateral_Surface"
2 4 "Top_Surface"
3 1 "Soil"
$EndPhysicalNames
..............

Figure 207.11 shows the geometry and mesh visualization in Gmsh. It is noted that Gmsh performs meshing for linear interpolation elements by default. In other words, the above cubic block geometry object is meshed into eight-node bricks, that have linear isoparametric interpolation, 8NodeBrick. For higher order interpolation meshing options, that is for meshing twenty-seven node brick elements mesh, 27NodeBrick for example, additional -order int should be used. int here is the integer specifying the
order of meshing. For example, the following terminal command with `-order 2`, i.e., 2\textsuperscript{nd} order meshing, generates twenty-seven node brick meshes (27NodeBrick):

```
gmsh Example_1.geo -o -order 2
```

### 207.2.6.3 Writing all gmESSI Commands for the model

Using gmESSI for mesh conversion is very easy. To achieve this, a [Example_1.gmessi] file is created containing all the required gmESSI commands to be executed sequentially. Let us look at each of them.

Since physical group names and ids are required for referring the gmESSI commands, its always best to copy all the physical group data from the .msh file (in this case [Example_1.msh] file) in the header of .gmessi file, so that its easier for th user to refer to the physical groups while writing commands in .gmessi file. The contents of the .gmessi file are shown below.

```
$cat Example_1.gmessi

### Physical Groups defined in the msh file.
#2 2 "Base_Surface"
#2 3 "Lateral_Surface"
#2 4 "Top_Surface"
#3 1 "Soil"

### loading the gmsh file
```
10 gmESSI.loadGmshFile("Example_1.msh")
11
### Defining the Simulation Directory and node, element, load and main file
### Its important to define the directory and these files at the beginning of any gmESSI command conversion
### 1 refers as overwrite mode (will overwrite the directory if present) --- 0 would not overwrite
15 gmESSI.setSimulationDir("./Example_1_ESSI_Simulation",1)
16 gmESSI.setMainFile(gmESSI.SimulationDir+ "main.fei")
17 gmESSI.setNodeFile(gmESSI.SimulationDir+ "node.fei")
18 gmESSI.setElementFile(gmESSI.SimulationDir+ "element.fei")
19 gmESSI.setLoadFile(gmESSI.SimulationDir+ "load.fei")

21 // is used to provide commands and gets translated in the main.fei file
22 Also, the commands followed by exclamation '!' get directly copied to the main.fei file
23 Usually, the user would write Real-ESSI DSL against the exclamation mark.

26 // My new model
27 ! model name "Soil_Block";
28 [Add_All_Node{ unit:= m, nof_dofs:= 3}]

31 // Adding Material also assigning it to elements
32 ! add material #1 type linear_elastic_isotropic_3d_LT mass_density = 2000*kg/m^3 elastic_modulus = 200*MPa poisson_ratio = 0.3;
33 [Add_8NodeBrick{Physical_Group#Soil, MaterialNo:= 1}]

37 ! include "node.fei";
38 ! include "element.fei";
39 ! new loading stage "Stage1_Self_Weight";

43 # Applying Fixities
46 [Fix_Dofs{Physical_Group#Base_Surface, all}]
47 [Fix_Dofs{Physical_Group#Lateral_Surface, ux uy}]

50 #### For applying Self-Weight Load to the soil elements
51 ! add acceleration field #1 ax = 0*g ay = 0*g az = -1*g;
52 ! add load #1 to all elements type self_weight use acceleration field #1;

59 #Updating the tag inside gmESSI as user entered by himself load tag
60 gmESSI.setESSITag("load",2)

67 ! include "load.fei";
68 ! NumStep = 10;

71 ! define algorithm With_no_convergence_check;
72 ! define solver UMFPack;
73 ! define load factor increment 1/NumStep;
74 ! simulate NumStep steps using static algorithm;
### updating the new load file before new loading stage
```
gmESSI.setLoadFile(gmESSI.SimulationDir+ "Surface_Load.fei")
```

### For applying Surface load on the Top Surface of the Soil Block
```
[Add_8NodeBrick_SurfaceLoad{Physical_Group#Soil,Physical_Group#Top_Surface,10*Pa}]
```

##### For applying Nodal loads to all the nodes of the top surface
```
#[Add_Node_Load_Linear{Physical_Group#Top_Surface, ForceType:= Fx, Mag:= 10*kN}]
```

### For applying Surface load on the Top Surface of the Soil Block
```
! include "Surface_Load.fei";
```

### Executing gmESSI on Example 1.gmessi input file

Once .gmessi input file is ready, the next task is to run it using the ‘gmessy’ command in terminal. Running would carryout the translation to all and produce the log of translation, displayed on the terminal.

```
$ gmessy Example_1.gmessi
Message:: newDirectory created as ./Example_1_ESSI_Simulation
Add_All_Node{ unit:= m, nof_dofs:= 3} Found!! Successfully Converted
Add_8NodeBrick{Physical_Group#Soil, MaterialNo:= 1} Found!! Successfully Converted
Fix_Dofs{Physical_Group#Base_Surface, all} Found!! Successfully Converted
```
Fix_Dofs{Physical_Group#Lateral_Surface, ux uy}
  Found!!
  Successfully Converted

Add_8NodeBrick_SurfaceLoad{Physical_Group#Soil,Physical_Group#Top_Surface,10*Pa}
  Found!!
  Successfully Converted

************************ Updated New Tag Numbering ***********************

damping = 1
displacement = 1
element = 126
field = 1
load = 27
material = 1
motion = 126
node = 217
nodes = 217
Gmsh_Elements = 276
Gmsh_Nodes = 217

It would create a folder [Example_5_ESSI_Simulation] and places load.fei, node.fei, element.fei and main.fei files. The user at this point do not need to write anything in the Example_5_analysis.fei file as every command was sequentially written down in .gmessi file and is converted. The content of the main.fei is shown below.

```
$cat Example_5_analysis.fei
// My new model
model name "Soil_Block";

// Adding Material also assigning it to elements
add material #1 type linear_elastic_isotropic_3d_LT mass_density = 2000*kg/m^3 ↔
esthetic_modulus = 200*MPa poisson_ratio = 0.3;

include "node.fei";
include "element.fei";
new loading stage "Stage1_Self_Weight";

add acceleration field # 1 ax = 0*g ay = 0*g az = -1*g ;
add load #1 to all elements type self_weight use acceleration field # 1;
include "load.fei";
NumStep = 10;

define algorithm With_no_convergence_check;
define solver UMFPack;
define load factor increment 1/NumStep;
simulate NumStep steps using static algorithm;
```
22 new loading stage "Stage2_Surface_Loading";
23 include "Surface_Load.fei";
24 NumStep = 10;
25
26 define algorithm With_no_convergence_check;
27 define solver UMFPack;
28 define load factor increment 1/NumStep;
29 simulate NumStep steps using static algorithm;
30 bye;

207.2.6.5 Running Real-ESSI and visualization in paraview

With all files ready in their place, the next step is to run the main.fei file directly in ESSI.

```
$essi -f main.fei
```

Running ESSI creates .feioutput file which can be visualized in paraview using PVESSIReader plugin. Figure ?? shows the visualization of hdf5 output produced in paraview.
Figure 207.12: Visualizing output in Paraview.
207.2.7 Illustrative Examples

The Examples directory of gmESSI folder contains five examples as Example_1, Example_2.... and Example_5. They are summarized as

1. **[Example_1]**: Modeling of Surface load on block of Soil. The geometry (.geo), mesh (.msh) and .gmessi input files can be downloaded HERE.

2. **[Example_2]**: Modeling of Cantilever Beam. The geometry (.geo), mesh (.msh) and .gmessi input files can be downloaded HERE.

3. **[Example_3]**: Modeling of Tower (beam) located above the ground and embedded in soil using contact/interface elements. The geometry (.geo), mesh (.msh) and .gmessi input files can be downloaded HERE.

4. **[Example_4]**: Modeling of a concrete foundation on Soil connected by contact elements. The geometry (.geo), mesh (.msh) and .gmessi input files can be downloaded HERE.

5. **[Example_5]**: Modeling of a embedded shells and beam in Solids. The geometry (.geo), mesh (.msh) and .gmessi input files can be downloaded HERE.

[Example_1] was discussed in the previous section. Examples 1 to 4 are discussed and refereed in the manual at several instances. The user is encouraged to over these examples and learn to create geometry '.geo' and .gmessi input files. Here, two examples Example_2 about cantilever beam analysis and Example_5 about beams and shell is discussed.
207.2.7.1 Modeling of Cantilever Beam With Surface Load [Example_2]

![Illustration of the cantilever problem.](image)

The problem consist of a cantilever beam with its left end fixed. A uniform surface load of $P$ is applied. The geometry (.geo) and the gmessi input file for this problem can be downloaded here. Figure 207.14 shows visualization of output after running the model in Real-ESSI.

![Illustration of the cantilever problem.](image)
207.2.7.2 Modeling of a embedded shells and beam in Solids [Example.5]

The problem consist of solid of 3 dofs in which beams and shells of 6dofs are embedded. The embedded beams and shell elements are connected by contact/interface elements. A nodal load to the top of the beam and shell is applied. The geometry (.geo) and the gmessi input file for this problem can be downloaded here. Figure 207.16 shows visualization of output after running the model in Real-ESSI.

Figure 207.15: Illustration of the cantilever problem.
Figure 207.16: Visualizing displacement field in Paraview.
207.2.8 Realistic Models Developed Using gmESSI

Figure 207.17: Nuclear Power Plant model 1, half model shown, with vertical plane cut.

207.3 Introduction to SASSI-ESSI Translator

This section will cover a simple mesh translator that translates mesh from SASSI format into ESSI format.
Figure 207.18: Nuclear Power Plant model 2, half model shown, with vertical plane cut.
Figure 207.19: Shear Box.
Chapter 208

Real-ESSI Post Processing Methods

(In collaboration with Mr. Sumeet Kumar Sinha, Dr. Yuan Feng, Dr. Han Yang, and Dr. Hexiang Wang)
208.1 Introduction
208.2 Model Results Post-Processing

This chapter describes methodology for post processing simulation results from the Real-ESSI Simulator. Two main approaches are used:

- Plotting time histories of scalar results (described in section 208.3 on page 1246), using Python and/or Matlab for:
  - components of displacements, velocities, accelerations, pore fluid pressures, for finite element nodes,
  - components of stress and/or strain at integration (Gauss) points within each finite element,
  - components of section forces for structural finite elements
  - energy input and dissipation for parts or whole of the volume/model, in incremental and/or cumulative form

- Visualization of a part or a complete model for displacements, velocities, accelerations, stress and strain components, sectional forces, energy dissipation through visualization system ParaView, as described in section 208.4 on page 1247.
208.3 Time Histories Plotting

Time histories of various scalar results (as listed above) can be extracted from output files, saved in HDF5 (Group, 2020), in a format described in chapter 206, on page 1136 in Jeremić et al. (1989-2021).
208.4 Post Processing and Visualization using ParaView

ParaView package http://www.paraview.org/ (Ayachit, 2015) is a powerful multi-platform data analysis and visualization application available as an Open Source.

It can be run on supercomputers to analyze datasets of peta-scale size as well as on laptops for smaller data, has become an integral tool in many national laboratories, universities and industry, and has won several awards related to high performance computation.

PVESSIReader is a plugin for paraview that integrates Real-ESSI Simulator output to Paraview for visualization. PVESSIReader reads Real-ESSI output file, in HDF5 format, files with extension .feioutput. The plugin works for sequential, parallel as well as remote visualization mode. It has a number of visualization features to visualize stresses, eigen modes, relative displacement, physical groups, energy dissipation, etc.

The installation of both Paraview and PVESSIReader is described in some detail in section 208 on page 1243 of the main document (Jeremić et al., 1989-2021).
208.4.1 Visualization in ParaView: Features

PVESSIReader has been consistently developed and added with lot of visualization options which are built on ParaView Visualization Toolkit (VTK) framework. This section shows all the visualization options that PVESSIReader offers other than what is available in ParaView. The features are illustrated in subsections below with help of examples in Examples folder of PVESSIReader source directory.

Before, looking at the features, it’s important to know how PVESSIReader works. PVESSIReader takes Real-ESSI HDF5 output (.feioutput) file format as input and creates a PVESSIReader folder inside the HDF5 file. PVESSIReader does this to ensure the visualization to be optimized. The contents inside this folder are not important for any regular user. The contents of ”PVESSIReader” folder is shown in Figure 208.1, although it is not important to regular users. User must know that, the plug-ins first creates this folder and then uses the content of this folder for visualization in ParaView. So creation/reading of this folder is the essence to visualization in ParaView.

![Contents of PVESSIReader folder.](image)

208.4.1.1 PVESSIReader Visualization Options

By default PVESSIReader builds a node mesh. Figure 208.2 shows the various visualization options that is available for PVESSIReader. The Gauss to node interpolation is turned off and other options shown in Figure 208.2 is turned off. As stated in previous Section 208.4.1, the plugin creates the PVESSIReader
folder inside the output HDF5 file only once and uses it for rest of visualization (even after you close and reopen it). The ‘Build PVESSIReader folder’ button shown in Figure 208.2 on clicking rebuilds the ‘PVESSIReader folder’. If in case the loading of '.feioutput' file fails in ParaView, the user should clicks the ‘Build PVESSIReader Folder’ before hitting ‘Apply’ button.

Description of various PVESSIReader visualization options in the order shown in Figure 208.2 is listed below.

- **Build PVESSIReader Folder** - Rebuild the content of PVESSIReader folder inside output file
- **Refresh** - It reloads the current visualization view.
- **Enable Gauss to Node Interpolation** - Enables the interpolation from Gauss points to node using shape functions. It works only for 8 node and 27 node brick with 8 and 27 Gauss points respectively. See Section 208.4.1.9.
- **Enable uPU Visualization** - Enables the visualization of fluid displacements (U) and pore-pressure (P) at nodes. See Section 208.4.1.10.
- **Enable Relative Displacement** - When this is enabled, the displacement of any time step can be visualized with respect to any other reference time step. See Section 208.4.1.6.

- **Reference Time Step No** - This option is used to set the reference time step number about which the relative displacements would be visualized. See Section 208.4.1.6.

- **Show Gauss Mesh** - This option can be enabled to visualize the Gauss points (mesh) of the entire model. See Section 208.4.1.8.

- **Enable Displacement Probing** - This option only works if Gauss mesh option is enabled. With this option, displacements are calculated at Gauss points using ParaView interpolation functions for each elements containing those Gauss points. See Section 208.4.1.8.

- **Physical Groups** - This option is enabled to visualize pre-defined physical groups in Real-ESSI input or manually defined selected nodes or elements. See Section 208.4.1.12.

- **Enable Actual Time Step Values** - By default instead of actual simulation time (in seconds), time step number of analysis is provided to ParaView VCR. Figure 208.3 shows the result of enable/disable of this option.

![Figure 208.3: Illustration of difference between enable and disable of actual time step values.](image)

In the Figure 208.3, the enabled option gives the exact simulation time of 3.895s. Whereas, disabling the option shows time step number of 200.

The visualization gets automatically updated on enable/disable of options. When one hit’s apply, the corresponding changed result gets updated i.e. the mesh would get real time updated with enable/disable of these options.
208.4.1.2 Sequential Visualization

Sequential visualization means visualizing the Real-ESSI output on a single core of laptop/desktop. This is used for single output files that Real-ESSI produces for sequential runs. Figure 208.4 shows the visualization of output file produced by sequential Real-ESSI simulation.

1. To open one core output in sequential

1   cd pvESSI/Examples
2   ParaView ShearBox_Parallel.h5.1.feioutput

The parallel output files of ESSI can also be visualized sequentially. Each individual (core) file can be sequentially visualized showing only a part of the model results. Also all the parallel files at once can be opened as well in ParaView as shown in Figure 208.5. All PVESSIReader examples can be obtained at http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_pvESSI/Examples.

1. To open one core output in sequential

1   cd pvESSI/Examples
2   ParaView ShearBox_Parallel.h5.1.feioutput

2. To open all cores output in parallel

1   cd pvESSI/Examples
2   ParaView ShearBox_Parallel.h5.feioutput

Figure 208.4: Sequential Visualization of output produced by sequential Real-ESSI simulation.
(a) Visualizing only one slave output file  

(b) Visualizing all slave output files at once

Figure 208.5: Sequential Visualization of output produced by parallel Real-ESSI simulation.
208.4.1.3 Remote Visualization

Remote visualization is an important feature that ParaView offers. This is an important feature in need, when simulations are run on super computers with thousands of cores. The steps for remote visualization are shown below with an example on local desktop.

1. Run `pvserver` on server

```bash
1 $pvserver
2 Waiting for client...
3 %Connection URL: cs://sumeet:11111
4 %Accepting connection(s): sumeet:11111
5 Connection URL: cs://jeremic:11111
6 Accepting connection(s): jeremic:11111
```

2. Open ParaView on client side and click on `connect` button located on top left window.

   ![Figure 208.6: Connect Server.](image)

3. Select and connect to the server and then load the plugins on both client and server side as shown in Figure 208.7

   ![Figure 208.7: Connect to the server and load plugins.](image)

4. Navigate to `pvESSI/Examples/ShearBox Parallel.h5.1.feiooutput` and hit apply

208.4.1.4 Parallel Visualization

Parallel visualization is similar to remote visualization. The only difference is to start the `pvserver` in parallel on multiple cores. It is recommended to have the same number of cores that was used in Real-ESSI parallel simulation. Below shows steps on how to do parallel visualization in ParaView.
The next following steps are same to that of Remote visualization as shown in Section 208.4.1.3. Thus, parallel visualization can be performed remotely as well as locally.

```bash
1 $ mpirun -np $(nop) pvserver
2 # $(nop) is replaced by number of cores on which parallel visualization is to be run.
```
208.4.1.5 General Field Visualization

Below is the list of general visualization variables available for any model in ParaView using PVESSIReader plugin. The following subsections describes each option through an example. The examples can be found in pvESSI/Examples directory. The example file 'ShearBox_Sequnential.h5.feioutput' can be downloaded here.

```
cd pvESSI/Examples
ParaView ShearBox_Sequnential.h5.feioutput
```

**Displacement Field**: The displacement field represents the total displacement from the start of the Real-ESSI simulation. There are two modes of displacement field visualization available in ParaView.

**NOTE**: Please remember to change step number from 0 to any other step number, as all output, including displacements, is 0.0 at step 0.

1. **Scalar field visualization**: The options available are each individual displacement vector components $u_x, u_y$ and $u_z$ in x,y,z directions respectively. It also shows the displacement magnitude $|u| = \sqrt{u_x^2 + u_y^2 + u_z^2}$. The units of displacements field in [m].

![Select Displacement Field](image1.png)

![Select scalar field](image2.png)

(a) Select Displacement Field  (b) Select scalar field

Figure 208.9: Displacement Scalar Field Visualization.

2. **Vector Field visualization**: This is achieved using 'Wrap by Vector' plugin available in ParaView. Figure 208.10 and Figure 208.11 shows steps to visualize deformed mesh.
Figure 208.10: Deformation Visualization.

Figure 208.11: Displacement field visualization in ParaView.
Boundary Conditions: Again this a vector field which contains information about boundary conditions i.e fixities applied in $u_x$, $u_y$ and $u_z$ directions. A value of 1 means the node is fixed while 0 means it is free. Figure 208.12 shows steps to visualize boundary conditions.

(a) Select Boundary Conditions

(b) Select Fixities Type

(c) Boundary Conditions in $u_z$

Figure 208.12: Boundary Conditions Visualization.
Material Tag Visualization: This is a scalar field visualization that shows the material tag associated with the elements in Real-ESSI simulation. Figure 208.13 shows steps to visualize element’s material tag.

![Select Material Tag](image1.png)
(a) Select Material Tag

![Material Tag Field](image2.png)
(b) Material Tag Field

Figure 208.13: Material Tag Visualization.
**Node Tag Visualization**: This is a scalar field visualization that shows the node tag associated with the nodes in Real-ESSI simulation. Figure 208.14 shows steps to visualize node’s tag.

![Node Tag Visualization](image)

(a) Select Node Tag  
(b) Node Tag Field

Figure 208.14: Node Tag Visualization.
Element Tag Visualization: This is a scalar field visualization that shows the element no associated with the elements in Real-ESSI simulation. Figure 208.15 shows steps to visualize element’s tag.

(a) Select Element Tag  
(b) Element Tag Field

Figure 208.15: Element Tag Visualization.
**Element Class Tag Visualization**: This is a scalar field visualization that shows the **Element’s Class Tag** number associated with each element type in Real-ESSI simulation. Figure 208.15 shows steps to visualize element’s tag. Section 206.5.4.4 shows the class tag for various element types available in Real-ESSI.

![Image of visualization](image-url)

(a) Select Class Tag

(b) Class Tag Field

Figure 208.16: Class Tag Visualization.
208.4.1.6 Relative Displacement Visualization

When the ‘Enable Relative Displacement’ is checked, the relative displacement visualization option becomes active. By default, the relative displacement time step number is set to '0' as shown in Figure 208.17. Time step number ‘0’ corresponds to initial conditions of the loading stage output file.

![Enable Relative Displacement](image)

Figure 208.17: Enable Relative Displacement.

**Reference Time Step Number** - It defines the relative time step index number for relative displacement visualization. By default it is set to 0 i.e. to the initial conditions. It is very useful, when one wants to visualize deformation coming from the stage itself. For example:- Separating self-weight from Static Pushover Analysis in the Shear Box simulation. The steps to do the same is shown below. The example file ShearBox_PushOver.h5.feioutput can be downloaded [here](link).

1. Open an example in ParaView

   ```
   cd pvESSI/Examples
   ParaView ShearBox_PushOver.h5.feioutput
   ```

2. Check on Enable Relative Displacement under visualization options

3. Apply *warp by vector* plugin and follow the steps as shown in Figure 208.10

Figure 208.18 shows the visualization with and without relative displacement.
Figure 208.18: Pushover analysis of Shear Box after self-weight load application.
208.4.1.7 Visualizing Element’s Partition

If the ESSI simulation was run in parallel mode, it becomes important to visualize the elements distribution between different cores. In ParaView, one can see the element distribution by selecting "Partition Info". Following is shown in Figure 208.19 an example to visualize mesh partitioning. All the example files can be obtained here.

```
1  cd pvESSI/Examples
2  ParaView ShearBox_Parallel.h5.feiooutput
```

and then select **Partition Info** as shown below in Figure 208.19

![Select Partition Info](image1.png)

![Mesh Partitioning](image2.png)

(a) Select Partition Info  
(b) Mesh Partitioning

**Figure 208.19:** Visualizing mesh partitioning.
208.4.1.8 Gauss Mesh Visualization Options

Often, it is required to visualize stress and strain fields. Since stress or strains are evaluated at Gauss points in 3-D elements, Gauss mesh is needed to visualize them. PVESSIReader offers option to visualize Gauss mesh and its fields.

- **Show Gauss Mesh** - Shows only Gauss mesh with Gauss attributes.

- **Enable Displacement Probing** - When this option is enabled, displacements are probed to the Gauss location. Its useful in the situation, when one wants to visualize the change in stress with deformation. With this as active, one can apply ‘warp by vector’ filter.

It must be noted that the Enable Displacement Probing options only works when Show Gauss Mesh mode is enabled. Figure 208.20 shows the steps to visualize Gauss mesh.

![Gauss Mesh Options and Fields](image)

Figure 208.20: Visualizing Gauss mesh and its fields.

The various fields that can be visualized in Gauss mesh mode as shown in Figure 208.20 are shown below.

- **Total Strain** $\epsilon$ : It defines the total strain from the start of the simulation. It has six independent component $\epsilon_{xx}, \epsilon_{xy}, \epsilon_{xz}, \epsilon_{yy}, \epsilon_{yz}$ and $\epsilon_{zz}$. The magnitude of the total stress in ParaView is defined as $\sqrt{\epsilon_{ij} : \epsilon_{ij}}$.

- **Total Plastic Strain** $\epsilon^{pl}$ : It defines the total plastic strain from the start of the simulation. It has six independent component $\epsilon^{pl}_{xx}, \epsilon^{pl}_{xy}, \epsilon^{pl}_{xz}, \epsilon^{pl}_{yy}, \epsilon^{pl}_{yz}$ and $\epsilon^{pl}_{zz}$. The magnitude of the total plastic strain in ParaView is defined as $\sqrt{\epsilon^{pl}_{ij} : \epsilon^{pl}_{ij}}$.

- **Total Effective Stress** $\sigma^{t}$ : It defines the total effective stress from the start of the simulation. It has six independent component $\sigma^{t}_{xx}, \sigma^{t}_{xy}, \sigma^{t}_{xz}, \sigma^{t}_{yy}, \sigma^{t}_{yz}$ and $\sigma^{t}_{zz}$. The magnitude of the total effective stress in ParaView is defined as $\sqrt{\sigma^{t}_{ij} : \sigma^{t}_{ij}}$. The unit of visualization is in $[Pa]$. 
• **Total Mean Effective Stress** \( p \): It defines the total mean of the effective stress \( \sigma' \) from the start of the simulation. It is defined as \( p = -\sigma'_{ii}/3 \) as described in Equation ???. The unit of visualization is in \([Pa]\).

• **Total Deviatoric Effective Stress** \( q \): It defines the deviatoric invariant of the total effective stress \( \sigma' \) from the start of the simulation. It is defined as \( q = \sqrt{3}J_2' \) as described in Equation ???. Where, \( J_2 \) is the second invariant of the deviatoric stress tensor \( s_{ij} = \sigma'_{ij} - \sigma'_{kk}/3\delta_{ij} \). The unit of visualization is in \([Pa]\).

• **Total Mean Plastic Strain** \( \epsilon_{p}^{pl} \): It defines the mean total plastic strain \( \epsilon^{pl} \) invariant from the start of the simulation. It is defined as \( \epsilon_{p}^{pl} = -\epsilon_{ii}^{pl}/3 \). This visualization parameter is unit-less.

• **Total Deviatoric Plastic Strain** \( \epsilon_{p}^{pl} \): It defines the deviatoric invariant of the total plastic strain \( \epsilon^{pl} \) from the start of the simulation. It is defined as \( \epsilon_{p}^{pl} = \sqrt{3}J_2' \). Where, \( J_2' \) is the second invariant of the deviatoric plastic strain tensor \( e_{ij}^{pl} = \epsilon_{ij}^{pl} - \epsilon_{kk}^{pl}/3\delta_{ij} \). This visualization parameter is unit-less.

1. Open an example in ParaView. All the example files can be obtained at [http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_pvESSI/Examples](http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_pvESSI/Examples).

```plaintext
1 cd pvESSI/Examples
2 ParaView ShearBox_PushOver.h5.feioutput
```

2. Check on **Enable Relative Displacement** under PVESSIReader build options

3. Enable Gauss mesh as shown in Figure 208.20(a). Select **Mean Effective Stress** \( p \) [Pa]. The resulting visualization is shown in Figure 208.21(a).

4. Enable displacement probing as shown in Figure 208.20(a). Apply a warp by vector filter and select the vector displacement as shown in Figure 208.10. Now select again the **Mean Effective Stress** \( p \) [Pa] field option to visualize. The resulting visualization is shown in Figure 208.21(b).
Figure 208.21: Visualization of mean effective stress $p$ invariant in Gauss mesh.
208.4.1.9 Gauss To Node Interpolation Mode Visualization

This visualization mode can be enabled by checking the 'Gauss To Node Interpolation' option as shown in Figure 208.22(a). In this mode, the total effective stress $\sigma'_{ij}$, total strain $\epsilon_{ij}$, total plastic strain $\epsilon^p_{ij}$, total mean effective stress $p$, total deviatoric effective stress $q$, total mean plastic strain $\epsilon^p_p$ and total deviatoric plastic strain $\epsilon^q_q$ are interpolated from the Gauss points to the nodes of individual element. Individual shape functions of the element (with full Gauss integration) are used to obtain the stress or strain field at nodes. To smooth out the jumps in stress or strain field at the node by adjacent elements, unweighted averaging is performed. For the elements (usually structural) with no Gauss points, the stress or strain contribution at nodes are considered as zero. While taking the averaging, their contributions are not taken, as Real-ESSI does not output stress/strain for them.

In this mode, visualization of all the parameters listed and described in Section 208.4.1.8 is available. Figure 208.22 show the steps to enable and use Gauss to Node Interpolation option.

1. Open an example in ParaView. All the example files can be obtained at http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_pvESSI/Examples.

   ```
   cd pvESSI/Examples
   ParaView ShearBox_Sequential.feoutput
   ```

2. Follow the steps as shown in Figure 208.22

   **Note:** The option Gauss to node interpolation is provides only an approximate estimate for stress and strains at nodes. The values obtained at nodes is not accurate and thus Gauss Mesh Visualization option described in Section 208.4.1.8 must be performed to get the accurate stress and strains at Gauss points. Also, it must be noted that this option works only for 8 node brick with 8 Gauss points and 27 node brick with 27 node points. For elements which have less number of nodes that Gauss points, the total number of equations (unknowns) is not equal to constraints (knowns). In this case, only the shape function defined at the nodes are used to get the stress or strain back to the node.
(a) Enable Gauss To Node Visualization Option  
(b) Select the visualization variable

(c) Visualization of Deviatoric Stress interpolated to nodes

Figure 208.22: Steps to visualize stress and strain interpolated from Gauss points to nodes.
208.4.1.10 upU Visualization

This mode is to visualize the upU elements used in Real-ESSI simulation. Enabling this mode, produces additional outputs of 'Pore Pressure $p[Pa]$' and 'Fluid Displacement $U_x[m]$, $U_y[m]$ and $U_z[m]$' at nodes. These additional outputs are described below.

- **Pore Pressure $p[Pa]$**: It defines the pore-fluid pressure in the upU element at the nodes. The magnitude of the pore pressure is $[Pa]$.

- **Fluid Displacement $U[m]$**: It defines the displacement by the fluid particles of upU at nodes. The units is in meters $[m]$. The solid displacement is termed as $u$ and refers to the 'Displacement $u$' variable in visualization as described in Section 208.4.1.5.

Since general dry elements does not have any fluid, enabling this option would produce 'zero' pore fluid pressure and fluid displacements at nodes. Below is shown an example that shows how to use the upU visualization feature. Figure 208.23 shows the steps.

1. Open an example in ParaView. All the example files can be obtained at http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_pvESSI/Examples.

```
1   cd pvESSI/Examples
2   ParaView upU_Visualization_Example.feioutput
```

2. Follow the steps as shown in Figure 208.23
(a) Enable upU Mode Visualization Option

(b) Select the visualization variable either pore pressure or fluid displacement

(c) Visualization of Pore Pressure $p$ at nodes

Figure 208.23: Steps to visualize pore pressure $p$ or fluid displacements $U$ in upU visualization mode.
208.4.1.11 Eigen Mode Visualization

Visualization of eigen modes is that same as visualizing "displacements" and applying "warp by vector" filter on Eigen Value Analysis output of Real-ESSI simulation.

1. Open an eigen value analysis output. All the example files can be obtained at http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_pvESSI/Examples.

```
1 cd pvESSI/Examples
2 ParaView ShearBoxWall_Eigen_Analysis.h5.feioutput
```

2. Select displacement field and then apply warp by vector plugin and selected its properties

(a) Select Displacement Field
(b) Select plugin
(c) Plugin properties

Figure 208.24: Eigen Modes visualization.

3. Now $n^{th}$ time steps here, corresponds to the $n^{th}$ eigen mode.

(a) Eigen Mode 5
(b) Eigen Mode 9

Figure 208.25: Few eigen modes.
208.4.1.12 Visualizing Physical Node and Element groups

In Real-ESSI it is possible to define different physical groups, for nodes and for elements. If one has defined physical groups in Real-ESSI, you can visualize the same in ParaView. There are two sections here that shows all the physical groups (nodes and elements) defined in the model as shown in Figure 208.26. Section 205.3.4.36 and Section 205.3.4.43 shows how to define and add physical group of nodes and elements respectively in Real-ESSI. All the example files can be obtained at http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_pvESSI/Examples.

```
1 cd pvESSI/Examples
2 ParaView Model-With_Physical_Groups.h5.feioutput
```

![Figure 208.26: Physical Group Visualization Options.](image)

This feature is very useful, when one is interested only in some specific regions of the model than the whole model. Also, This feature becomes very useful, for complicated "interested region/parts of the mesh", which cannot be selected by usual clip/box/..etc filters

1. **Enable Physical Element Group Selection** - Enables the selection of Physical Element Group. By default, it is disabled and one would see the whole mesh. By enabling it, one would only see the selected ‘Physical Element Groups’.

2. **Physical Element Groups** - It shows all the physical element groups defined in Input file of Real-ESSI. The user can select (one or more) of physical groups and hit apply to visualize them. It would show any effect only if the above options **Enable Physical Element Group Selection** is checked. Figure 208.27 shows steps to visualize the physical element groups defined in input files.
using Real-ESSI DSL.

(a) Select ‘Physical Element Group Selection’ Option  (b) ElementGroup : Auxiliary Building Interior Walls

Figure 208.27: Visualization of physical groups predefined in input file using Real-ESSI DSL.

3. **Element Tags** - This option provides user and interface to manually write element tags to be visualized. The user should enter the element tags against this option as a integer list separated by space. For example: ‘2 10 12 13 16’, where each of the number corresponds to the element tag defined in the model. Again, this option would only work if **Enable Physical Element Group Selection** option is checked. Figure 208.28 shows steps to visualize elements defined manually.

(a) Select ‘Physical Element Group Selection’ Option  (b) ElementGroup : Manually Defined Element Tags

Figure 208.28: Visualization of physical element group manually defined using PVESSEIReader option.
4. **Enable Physical Node Group Selection** - Enables the selection of Physical Node Group. By default, it is disabled and you would see the whole mesh. By enabling it you would only see the selected Physical Node Groups.

5. **Physical Node Groups** - It shows all the physical node groups defined in Input file of Real-ESSI. The user can select (one or more) of physical groups and hit apply to visualize them. It would show any effect only if the above options **Enable Physical Node Group Selection** is checked. Figure 208.29 shows steps to visualize the physical element groups defined in input files using Real-ESSI DSL.

![Figure 208.29](image)

(a) Select ‘Physical Node Group Selection’ Option

(b) NodeGroup : Auxiliary Building Interior Walls

Figure 208.29: Visualization of physical groups predefined in input file using Real-ESSI DSL.

6. **Node Tags** - This options provides user and interface to manually write node tags to be visualized. The user should enter the node tags against this option as a integer list separated by space. For example: ‘2 10 12 13 16’, where each of the number corresponds to the node tag defined in the model. Again, this option would only work if **Enable Physical Node Group Selection** option is checked. Figure 208.30 shows steps to visualize nodes defined manually.

**NOTE:** The user can also select both at once, i.e physical element group and physical node group, from the above menu. Figure 208.31 shows mixed selection.
(a) Select ‘Physical Node Group Selection’ Option

(b) NodeGroup : Manually Defined Node Tags

Figure 208.30: Visualization of physical node group manually defined using PVESSIReader option.

(a) ElementGroup : Auxiliary Building Interior Walls
(b) Node Group : Containment Cylinder
(c) Mixed Group : Containment Cylinder with Dome

Figure 208.31: Visualization of physical groups.
208.4.1.13 Using Threshold to Visualize Certain Elements

ParaView allows user to choose specific element types and only visualize selected elements. This function is achieved using Threshold. As shown in Figure 208.32, first click on the Threshold button in toolbar. Then, choose Element Class Tag in the drop-down list of Scalars, which can be found in Properties. A certain range of Element Class Tag can be chosen by setting the minimum and maximum values. If the minimum and maximum values are the same, only one element type will be selected and visualized.

![Click on Threshold](image1)

![Choose Element Class Tag](image2)

Figure 208.32: Using Threshold to Visualize Certain Elements.
A list of available element class tags in Real-ESSI is provided in Table 208.1.

Table 208.1: Available element class tags in Real-ESSI.

<table>
<thead>
<tr>
<th>Finite Element Name</th>
<th>Element Class Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truss Element</td>
<td>88</td>
</tr>
<tr>
<td>Shear Beam Element</td>
<td>93</td>
</tr>
<tr>
<td>Elastic Beam-Column Element</td>
<td>89</td>
</tr>
<tr>
<td>Timoshenko Elastic Beam-Column Element</td>
<td>129</td>
</tr>
<tr>
<td>Elastic Beam-Column Element with Lumped Mass</td>
<td>90</td>
</tr>
<tr>
<td>3D Displacement Based Fiber Beam-Column Element</td>
<td>95</td>
</tr>
<tr>
<td>4 Node ANDES Shell with Drilling DOFs</td>
<td>92</td>
</tr>
<tr>
<td>3 Node ANDES Shell with Drilling DOFs</td>
<td>91</td>
</tr>
<tr>
<td>Super Element Linear Elastic Import</td>
<td>9904</td>
</tr>
<tr>
<td>8 Node Brick Element (Order One, Two, Three, Four, Five, Six)</td>
<td>2 (14, 26, 38, 50, 62, 74)</td>
</tr>
<tr>
<td>8 Node Brick u-p Element (Order One, Two, Three, Four, Five, Six)</td>
<td>3 (15, 27, 39, 51, 53, 75)</td>
</tr>
<tr>
<td>8 Node Brick u-p-U Element (Order One, Two, Three, Four, Five, Six)</td>
<td>4 (16, 28, 40, 52, 64, 76)</td>
</tr>
<tr>
<td>20 Node Brick Element (Order One, Two, Three, Four, Five, Six)</td>
<td>5 (17, 29, 41, 53, 65, 77)</td>
</tr>
<tr>
<td>20 Node Brick u-p Element (Order One, Two, Three, Four, Five, Six)</td>
<td>6 (18, 30, 42, 54, 66, 78)</td>
</tr>
<tr>
<td>20 Node Brick u-p-U Element (Order One, Two, Three, Four, Five, Six)</td>
<td>7 (19, 31, 43, 55, 67, 79)</td>
</tr>
<tr>
<td>27 Node Brick Element (Order One, Two, Three, Four, Five, Six)</td>
<td>8 (20, 32, 44, 56, 68, 70)</td>
</tr>
<tr>
<td>27 Node Brick u-p Element (Order One, Two, Three, Four, Five, Six)</td>
<td>9 (21, 33, 45, 57, 69, 81)</td>
</tr>
<tr>
<td>27 Node Brick u-p-U Element (Order One, Two, Three, Four, Five, Six)</td>
<td>10 (22, 34, 46, 58, 70, 82)</td>
</tr>
<tr>
<td>Variable Node Brick Element (Order One, Two, Three, Four, Five, Six)</td>
<td>11 (23, 35, 47, 59, 71, 83)</td>
</tr>
<tr>
<td>Variable Node Brick u-p Element (Order One, Two, Three, Four, Five, Six)</td>
<td>12 (24, 36, 48, 60, 72, 84)</td>
</tr>
<tr>
<td>Variable Node Brick u-p-U Element (Order One, Two, Three, Four, Five, Six)</td>
<td>13 (25, 37, 49, 61, 73, 85)</td>
</tr>
<tr>
<td>8 Node Cosserat Brick Element</td>
<td>96</td>
</tr>
<tr>
<td>Bonded Contact/Interface/Joint Element</td>
<td>102</td>
</tr>
<tr>
<td>Force Based Dry Hard Contact/Interface/Joint Element</td>
<td>86</td>
</tr>
<tr>
<td>Force Based Dry Soft Contact/Interface/Joint Element</td>
<td>87</td>
</tr>
<tr>
<td>Force Based Coupled Hard Contact/Interface/Joint Element</td>
<td>97</td>
</tr>
<tr>
<td>Force Based Coupled Soft Contact/Interface/Joint Element</td>
<td>98</td>
</tr>
<tr>
<td>Stress Based Dry Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>99</td>
</tr>
<tr>
<td>Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior</td>
<td>100</td>
</tr>
<tr>
<td>Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior</td>
<td>101</td>
</tr>
<tr>
<td>Stress Based Dry Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>107</td>
</tr>
<tr>
<td>Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior</td>
<td>108</td>
</tr>
<tr>
<td>Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior</td>
<td>109</td>
</tr>
<tr>
<td>Stress Based Coupled Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>104</td>
</tr>
<tr>
<td>Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior</td>
<td>105</td>
</tr>
<tr>
<td>Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior</td>
<td>106</td>
</tr>
<tr>
<td>Stress Based Coupled Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior</td>
<td>110</td>
</tr>
<tr>
<td>Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior</td>
<td>111</td>
</tr>
<tr>
<td>Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior</td>
<td>112</td>
</tr>
</tbody>
</table>
(a) Full Model

(b) Only Fiber Beam-Column Elements

Figure 208.33: An example of using threshold to visualize fiber beam-column elements in a soil-structure model.
Chapter 209

Software Platform Build Process

(In collaboration with Prof. José Abell, Dr. Yuan Feng, Dr. Hexiang Wang, and Dr. Han Yang)
209.1 Chapter Summary and Highlights

209.2 Introduction to the Real-ESSI Simulator Program

The Real-ESSI Simulator systems consists of the Real-ESSI Program, Real-ESSI Computer and Real-ESSI Notes. Alternative name for the Real-ESSI Simulator system is Real-ESSI Simulator system. Pronunciation of Real-ESSI is similar to "real easy", as in "as easy as pie", while translation of Real ESSI to other languages (languages of developers and users) is also available: Врло просто, Просто κ’ο pasuł, Muy fácil, Molto facile, अच्छा सहज, 本当に簡単, Πραγματικά εύκολο, बहुत ही आसान, آسان واقعی, Très facile, Вистински лесно, Wirklich einfach, سهل جدا.

209.3 Real-ESSI Program Executables Download and Install

Executables for the Real-ESSI Simulator program are available online. Pre-built executables are available for Linux, Ubuntu 18.04, and can be downloaded and installed by analyst.

In order for prebuild executables to be able to run on a user/analyst computer, system libraries have to be brought up to date and addtional libraries installed. System libraries update/upgrade:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
sudo apt-get autoremove
```

For sequential and/or parallel version of Real-ESSI, additional libraries are needed, as described below:

209.3.1 Sequential Version of Real-ESSI Program

Libraries required to be installed for using sequential version of the Real ESSI program:

```
sudo apt-get install libboost-all-dev
sudo apt-get install libhdf5-dev
sudo apt-get install libtbb-dev
```

209.3.2 Parallel Version of Real-ESSI Program

Libraries required to be installed for executing parallel version of the Real ESSI program:

```
sudo apt-get install libboost-all-dev
sudo apt-get install libhdf5-dev
sudo apt-get install libtbb-dev
sudo apt-get install mpich
```
209.3.3 Real-ESSI Executable Downloads

The Real-ESSI program executables can be downloaded from Real-ESSI Simulator web site: http://real-essi.info/. Alternatively, contact Prof. Jeremić to arrange for customized Real-ESSI executables.

209.4 Real-ESSI Simulator System Install

In addition to the Real-ESSI Program, Real-ESSI Simulator system consists of a pre-processing modules and post-processing modules. Installation of pre-processing modules is described in Chapter 207, on page 1176 of the main document, lecture notes (Jeremić et al., 1989-2021). Installation of post-processing modules is described in Chapter 208, on page 1243 of the main document, lecture notes (Jeremić et al., 1989-2021).

Both pre and post processing manuals are also available through the main Real-ESSI Simulator web site: http://real-essi.info/.

209.5 Quick Build Procedures for Sequential and Parallel Versions of the Real-ESSI Program

These build procedures are meant for users that have access to Real-ESSI Program source code. Procedures assume Ubuntu version 16.04, or 18.04.

209.5.1 Update Compiler for Ubuntu 16.04, while 18.04 already has the latest version

Ubuntu 16.04 does not have the gcc-7 and thus this new version of a compiler needs to be installed needs to be installed.

```
1  sudo apt-get install build-essential
2  sudo add-apt-repository ppa:ubuntu-toolchain-r/test
3  sudo apt-get update
4  sudo apt-get install gcc-7 g++-7 cpp-7 gfortran-7 cmake
```

To change the default version of compilers, that executes with a call to gcc command, we have to configure the alternatives in the system. First remove any previous configuration with:

```
1  sudo update-alternatives --remove-all gcc
```
This might produce an error: `update-alternatives: error: no alternatives for gcc` if no previous alternatives were configured. Pay no attention to this error and continue. Alternatives will be configured according to the versions available on the computer. Each alternative will have a priority associated with it. When a link group is in automatic mode, the alternatives pointed to by members of the group will be those which have the highest priority. To configure alternatives, execute:

```
1 sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 90 --slave /usr/bin/g++ g++ /usr/bin/g++-7 --slave /usr/bin/gfortran gfortran /usr/bin/gfortran-7 --slave /usr/bin/cpp cpp /usr/bin/cpp-7
```

It does help to verify that proper compiler version is installed:

```
1 gcc --version
2 g++ --version
3 cpp --version
4 gfortran --version
```

### 209.5.2 Sequential Real-ESSI Build

**209.5.2.1 Libraries Required for Building Dependencies for Ubuntu 16.04 and 18.04.**

```
1 sudo apt-get install -y cmake
2 sudo apt-get install build-essential
3 sudo apt-get install flex
4 sudo apt-get install bison
5 sudo apt-get install libboost-all-dev
6 sudo apt-get install libtbb-dev
7 sudo apt-get install valgrind
8 sudo apt-get install libopenblas-dev
9 sudo apt-get install liblapack-dev
10 sudo apt-get install libpthread-workqueue-dev
11 sudo apt-get install zlib-dev
12 sudo apt-get install libssl-dev
13 sudo apt-get install mpich
14 sudo apt-get install libopenmpi-dev
```

It is noted that Real-ESSI requires libboost version 1.70 or above. For ubuntu 16.04, the system default libboost from `sudo apt-get install` might not meet this requirement.

Check the version of libboost by running:
1 cat /usr/include/boost/version.hpp | grep BOOST_LIB_VERSION

If the output version number is lower than 1.70. A newer version that meets the above requirement can be manually installed by running:

1 sudo rm -rf /usr/include/boost
2 wget https://dl.bintray.com/boostorg/release/1.70.0/source/boost_1_70_0.tar.gz
3 tar -xzvf boost_*.*.tar.gz
4 cd boost_*
5 sudo cp -r boost /usr/include/

### 209.5.2.2 Libraries Required for Building Utilities for Ubuntu 16.04 and 18.04.

1 sudo apt-get install liboctave-dev
2 sudo apt-get install libhdf5-serial-dev
3 sudo apt-get install hdf5-tools
4 sudo ln -sf /usr/lib/x86_64-linux-gnu/libhdf5_serial.so /usr/lib/libhdf5.so
5 sudo ln -sf /usr/lib/x86_64-linux-gnu/libhdf5_serial_hl.so /usr/lib/libhdf5_hl.so
6 sudo apt-get install libphonon-dev
7 sudo apt-get install libphonon4
8 sudo apt-get install qt4-dev-tools
9 sudo apt-get install qt4-qmake
10 sudo apt-get install libqt4-dev
11 sudo apt-get install libqt4-opengl-dev
12 sudo apt-get install mesa-common-dev
13 sudo apt-get install python-dev
14 sudo apt-get install python-numpy
15 sudo apt-get install python-matplotlib
16 sudo apt-get install python-scipy

### 209.5.2.3 Obtain Real-ESSI Sources

Make a directory where all the sources will reside and go there:

1 mkdir RealESSI_ROOT/
2 cd RealESSI_ROOT/

Obtain Real-ESSI sources from the github:

1 mkdir RealESSI_ROOT
2 git reset --hard
3 git pull

or alternatively:

1 mkdir RealESSI_ROOT
# using curly brackets to help in checking scripts, that rely on these brackets being available around URL

```bash
git clone {https://github.com/BorisJeremic/Real-ESSI.git} # Need permission → from Boris Jeremic for Real-ESSI on github
```

if you have access to the Real-ESSI program archived sources, copy them here and unpack the archive (replace _archive_name_.tgz with the actual archive name, for example _Real-ESSI_Complete.03_Feb_2019_15h_16m_.tgz):

```bash
tar -xvzf _archive_name_.tgz
```

or, for the example above:

```bash
tar -xvzf _Real-ESSI_Complete.03_Feb_2019_15h_16m_22s__Sunday.tgz
```

Go to the Real-ESSI source directory:

```bash
cd Real-ESSI/
```

Download dependencies:

```bash
mkdir -p ../RealESSI_Dependencies
mkdir -p ../RealESSI_Dependencies/include
mkdir -p ../RealESSI_Dependencies/lib
mkdir -p ../RealESSI_Dependencies/bin
mkdir -p ../RealESSI_Dependencies/SRC
cd ../RealESSI_Dependencies
#
rm -rf Dependencies_SRC.tar.gz
#
# using curly brackets to help in checking scripts, that rely on these brackets being available around URL
# wget {http://sokocalo.engr.ucdavis.edu/~jeremic/RealESSI/Dependencies_SRC.tar.gz}
#
tar -xzvf ./Dependencies_SRC.tar.gz -C ./SRC --strip-components 1
#
cd ../Real-ESSI
```

Then, download utilities:

```bash
mkdir -p ../RealESSI_Utilities
mkdir -p ../RealESSI_Utilities/include
mkdir -p ../RealESSI_Utilities/lib
mkdir -p ../RealESSI_Utilities/bin
mkdir -p ../RealESSI_Utilities/SRC
cd ../RealESSI_Utilities
#
8  rm -rf Utilities_SRC.tar.gz
9  #
10  # using curly brackets to help in checking scripts, that rely on these
11  # brackets being available around URL
12  #
13  wget {http://sokocalo.engr.ucdavis.edu/~jeremic/RealESSI/Utilities_SRC.tar.gz}
14  #
15  tar -xzvf ./Utilities_SRC.tar.gz -C ./SRC --strip-components 1
16  #
17  cd ../Real-ESSI

209.5.2.4 Compile Real-ESSI Dependency Libraries

Start building dependency libraries, if needed:

1  time ./build_libraries suitesparse
2  time ./build_libraries arpack
3  time ./build_libraries hdf5_sequential
4  time ./build_libraries tbb
5  time ./build_libraries lapack
6  time ./build_libraries parmetis

Check that all prerequisite libraries are built

1  time ./build_libraries check_sequential

209.5.2.5 Compile and link Real-ESSI Program

Create directories for the main Real-ESSI build program:

1  mkdir bin
2  mkdir lib
3  rm -f -r build_sequential
4  mkdir build_sequential
5  cd build_sequential

Build and install the executable, using 16 CPUs in this case. Of course, if you have more CPUs available, you can use most of them.

1  time cmake ..
2  time make -j 16
3  make install

Rename essi to essi.sequential just so to distinguish it from the parallel executable:
1 cd ../bin
2 cp essi essi.sequential

Finally, install essi.sequential in system binary directory so that others can use it:

1 sudo rm /usr/bin/essi /usr/bin/essi.sequential
2 sudo cp essi.sequential /usr/bin/essi.sequential
3 sudo chmod a+x /usr/bin/essi.sequential

209.5.3 Parallel Real-ESSI Build

209.5.3.1 Libraries Required for Building Dependencies for Ubuntu 16.04 and 18.04.

1 sudo apt-get install -y cmake
2 sudo apt-get install build-essential
3 sudo apt-get install flex bison
4 sudo apt-get install libboost-all-dev
5 sudo apt-get install libtbb-dev
6 sudo apt-get install valgrind
7 sudo apt-get install libopenblas-dev
8 sudo apt-get install liblapack-dev
9 sudo apt-get install libpthread-workqueue-dev
10 sudo apt-get install libpthread-workqueue-dev
11 sudo apt-get install zlib1g-dev
12 sudo apt-get install libssl-dev
13 sudo apt-get install mpich
14 sudo apt-get install libopenmpi-dev

It is noted that Real-ESSI requires libboost version 1.70 or above. For ubuntu 16.04, the system default libboost from `sudo apt-get install` might not meet this requirement.

Check the version of libboost by running:

1 cat /usr/include/boost/version.hpp | grep BOOST_LIB_VERSION

If the output version number is lower than 1.70. A newer version that meets the above requirement can be manually installed by running:

1 sudo rm -rf /usr/include/boost
2 wget https://dl.bintray.com/boostorg/release/1.70.0/source/boost_1_70_0.tar.gz
3 tar -xzvf boost_*.*.tar.gz
4 cd boost_*
5 sudo cp -r boost /usr/include/

209.5.3.2 Libraries Required for Building Utilities for Ubuntu 16.04 and 18.04.
1. `sudo apt-get install liboctave-dev`
2. `sudo apt-get install libhdf5-serial-dev`
3. `sudo apt-get install hdf5-tools`
4. `sudo ln -sf /usr/lib/x86_64-linux-gnu/libhdf5_serial.so /usr/lib/libhdf5.so`
5. `sudo ln -sf /usr/lib/x86_64-linux-gnu/libhdf5_serial_hl.so /usr/lib/libhdf5_hl.so`
6. `sudo apt-get install libphonon-dev`
7. `sudo apt-get install libphonon4`
8. `sudo apt-get install qt4-dev-tools`
9. `sudo apt-get install qt4-qmake`
10. `sudo apt-get install libxt-dev`
11. `sudo apt-get install libqt4-opengl-dev`
12. `sudo apt-get install mesa-common-dev`
13. `sudo apt-get install python-dev`
14. `sudo apt-get install python-h5py`
15. `sudo apt-get install python-matplotlib`
16. `sudo apt-get install python-scipy`

### 209.5.3.3 Obtain Real-ESSI Sources

Make a directory where all the sources will reside and go there:

1. `mkdir RealESSI_ROOT/`
2. `cd RealESSI_ROOT/`

Copy Real-ESSI program archive to that location and unpack the archive (replace `_archive_name_.tgz` with the actual archive name, for example `_Real-ESSI_Complete.03_Feb_2019_15h_16m_22s__Sunday.tgz`):

1. `tar -xvzf _archive_name_.tgz`

or, for the example above:

1. `tar -xvzf _Real-ESSI_Complete.03_Feb_2019_15h_16m_22s__Sunday.tgz`

Go to the Real-ESSI source directory:

1. `cd Real-ESSI/`

Download dependencies:

1. `mkdir -p ../RealESSI_Dependencies`
2. `mkdir -p ../RealESSI_Dependencies/include`
3. `mkdir -p ../RealESSI_Dependencies/lib`
4. `mkdir -p ../RealESSI_Dependencies/bin`
5. `mkdir -p ../RealESSI_Dependencies/SRC`
6. `cd ../RealESSI_Dependencies`
7. `rm -rf Dependencies_SRC.tar.gz`
8. `#`

# using curly brackets to help in checking scripts, that rely on these 
# brackets being available around URL 

```
wget {http://sokocalo.engr.ucdavis.edu/~jeremic/RealESSI/Dependencies_SRC.tar.gz}
```

```
tar -xzvf ./Dependencies_SRC.tar.gz -C ./SRC --strip-components 1
```

```
cd ../Real-ESSI
```

Then, download utilities:

```
mkdir -p ../RealESSIUtilities
mkdir -p ../RealESSIUtilities/include
mkdir -p ../RealESSIUtilities/lib
mkdir -p ../RealESSIUtilities/bin
mkdir -p ../RealESSIUtilities/SRC
```

```
cd ../RealESSIUtilities
```

```
rm -rf Utilities_SRC.tar.gz
```

```
wget {http://sokocalo.engr.ucdavis.edu/~jeremic/RealESSI/Utilities_SRC.tar.gz}
```

```
tar -xzvf ./Utilities_SRC.tar.gz -C ./SRC --strip-components 1
```

```
cd ../Real-ESSI
```

### 209.5.3.4 Compile Real-ESSI Dependency Libraries

Start building dependency libraries:

```
time ./build_libraries petsc
```

```
time ./build_libraries initialize
```

```
time ./build_libraries hdf5_sequential
```

```
time ./build_libraries lapack
```

```
time ./build_libraries tbb
```

```
time ./build_libraries parmetis
```

Check that all prerequisite libraries are built

```
time ./build_libraries check_parallel
```
209.5.3.5 Compile and link Real-ESSI Program

Create directories for the main Real-ESSI build program:

```bash
1. mkdir bin
2. mkdir lib
3. rm -f -r build_parallel
4. mkdir build_parallel
5. cd build_parallel
```

Build and install the executable, using 16 CPUs in this case. Of course, if you have more CPUs available, you can use most of them.

```bash
1. time cmake -DCMAKE_CXX_COMPILER=/usr/bin/mpic++ -DPROGRAMMING_MODE=PARALLEL ..
2. time make -j 16
3. make install
```

Rename `essi` to `essi.sequential` just so to distinguish it from the parallel executable:

```bash
1. cd ../bin
2. cp essi essi.parallel
```

Finally, install `essi.sequential` in system binary directory so that others can use it:

```bash
1. sudo rm /usr/bin/essi /usr/bin/essi.parallel
2. sudo cp essi.parallel /usr/bin/essi.parallel
3. sudo chmod a+x /usr/bin/essi.parallel
```

209.6 Build Procedures for Sequential and Parallel Versions of the Real-ESSI Program

209.6.1 Libraries and Application Build Process

The Real-ESSI Program was designed and developed (primarily) for parallel, high performance computations, while a (secondary) sequential version is also available. Both version (same source code, small changes during compilation process, and different main application source code and compilation) are designed and developed (primarily) for the Real-ESSI Computer (or similar, distributed memory parallel computers).

Building Real-ESSI simulator requires some basic tools be present in the target system such as C++ (must support C++11 standard) and Fortran compilers, as well as some widely available libraries (BoostC++ (Abrahams and Gurtovoy, 2005; Ramey, 2005), and an MPI-2 implementation). We provide further dependencies needed by the program which must be compiled separately and are not version
Described below in some detail, are procedures that are necessary for compilation of Real-ESSI program/application. Since we use Ubuntu GNU/Linux system, our installation procedures are using Debian/Ubuntu syntax for installing libraries or additional sources, for example `apt-get install ...`. For Red-Hat based systems, one would use `yum install ...` or similar...

The build procedures for Real-ESSI and its dependencies and utilities are available for all the major Linux Ubuntu platforms. The sections below will go in depth about installing libraries and building dependencies, Real-ESSI and its utilities.

### 209.6.2 Installing Libraries

Currently the `gcc-7` compiler (or a version above) is recommended, although any standard C++ compiler can be used. The requirement is a compiler that fully supports the C++11 standard. Please note that a compiler other than GNU gcc or LLVM clang would require modification of makefiles and has not been tested. Some dependencies rely on the tool `cmake` for their build system. A development version of BOOST library ([http://www.boost.org/](http://www.boost.org/)) and TBB (Threading Building Blocks) library ([https://www.threadingbuildingblocks.org/](https://www.threadingbuildingblocks.org/)) also need to be installed (see below).

Previous sections provide up to date instructions for building Real-ESSI sequential and parallel version. Sections below provide some additional information that might be useful.

### 209.6.3 Obtaining Real-ESSI and Source organization

Real-ESSI program sources are available to developers/collaborators under a restrictive open source license. Real-ESSI sources can be cloned from github for the developers who have direct access to the code. The source code is placed in the Real-ESSI folder. For others, please contact Boris Jeremić for Real-ESSI distribution options.

```bash
mkdir RealESSI_ROOT
# # using curly brackets to help in checking scripts, that rely on these # # brackets being available around URL #
git clone {https://github.com/BorisJeremic/Real-ESSI.git} # Need permission ← from Boris Jeremic for Real-ESSI on github
```

If you are not interested in sources and just want to run the program, you will need to have linux installed with appropriate libraries available (as detailed below). Installation of the Real-ESSI Program is most optimal if remote users have available an up to date linux system (sequential and/or parallel) and if arrangements can be made for a temporary, simple/regular user, remote login (through a secure shell)
for Prof. Jeremic. This will allow us to compile and install all the necessary libraries and the executable. Distribution of executable without remote login is available also, however in this case we distribute an un-optimized (slow), version of the Real-ESSI Program.

Real-ESSI distribution must be organized in a folder structure like follows:

```
1 RealESSI_ROOT
  2 | 
  3 + Real-ESSI
  4 | 
  5 + RealESSI_Dependencies
  6 | 
  7 + RealESSI_Utilities
```

Where RealESSI_ROOT can be any directory within the filesystem, Real-ESSI contains the (version controlled) source of the main code, RealESSI_Dependencies contains the sources for the software Real-ESSI depends on (distributed in a tar-ball) and RealESSI_Utilities contains sources of the utilities that help make Real-ESSI really easy.

### 209.6.4 Real-ESSI Dependencies Build Process

Real-ESSI Simulator build process depends on three software sources:

1. Common software available through the Linux distribution (available through the distribution package manager) which we already installed, using guidance in the above section
2. Software libraries that Real-ESSI depends on (provided via a tar-ball), described in this section
3. Real-ESSI Simulator source code

The current release of Real-ESSI includes the following dependencies.

- **ATLAS 3.10.3** Provides an efficient BLAS implementation.
- **HDF5 1.8.17** For Real-ESSI output
- **LAPACK 3.6.1** Standard linear algebra suite
- **ParMETIS 4.0.3** Software for graph partitioning used in parallel Real-ESSI
- **PETSc 3.7.3** High performance, parallel suite of system of linear equations solvers used in parallel Real-ESSI.
- **SuiteSparse 4.5.3** Provides interfaces into system of equations linear solvers used in sequential Real-ESSI.
- **Blas 3.6.0** Basic Linear Algebra Subprograms providing standard building blocks for performing basic vector and matrix operations.

- **Cmake 3.7.0-rc2** Provides latest tools designed to build cmake and makefiles

Real-ESSI source comes up with **build_libraries** script that can build all the necessary dependencies required by Real-ESSI for both sequential and parallel case. The script is located in the `RealESSI_ROOT/Real-ESSI` folder. The build_libraries is a bash script which calls a makefile that has targets defined to build the required dependencies.

The first step is to download all the sources of dependencies that needs to be build. In addition a directory where Real-ESSI sources will be placed is to be created. To do this, one has to run

```
cd Real-ESSI
./build_libraries download
```

This would download all the libraries in `tar.gz` format and would place them in `/SRC` of `RealESSI_Dependencies` directory. The script accepts targets that can be used to build a particular library or all libraries at once. The available options to the scripts can be found by running the target help as shown below.

```
./build_libraries help
```

### Miscelleneous:
- list_dependencies: Lists all the available dependencies version from SRC folder
- list_build_dependencies: Lists all the dependencies library already build in lib folder

### Parallel:
- parallel: Builds all the necessary libraries for parallel Real-ESSI
- hdf5_parallel: Builds parallel hdf5
- petsc: Builds petsc
- clean_parallel: Cleans parallel libraries

### Sequential:
- sequential: Builds all the necessary libraries for sequential Real-ESSI
- parmetis: Builds parmetis andmetis
- suitesparse: Builds suitesparse
- arpack: Builds arpack
- hdf5_sequential: Builds sequential hdf5
- lapack: Builds lapack
- atlas: Builds tuned LAPACK and BLAS
- clean_sequential: Cleans sequential libraries

### Default:
Advanced users play with the various targets and their uses. Simple action is to build libraries one at a time, for a sequential version:

```bash
1. time ./build_libraries suitesparse  
2. time ./build_libraries arpack  
3. time ./build_libraries hdf5_sequential  
4. time ./build_libraries tbb  
5. time ./build_libraries lapack  
6. time ./build_libraries parmetis
```

or all at once for sequential version (it helps if they are build one at a time, to observe any potential compilation messages):

```bash
./build_libraries sequential
```

Similar procedures is used for building parallel libraries:

```bash
./build_libraries parallel
```

The user can also check whether all the libraries are successfully built:

```bash
1. ./build_libraries check_sequential  
2. ./build_libraries check_parallel
```

Before building libraries one needs to follow procedure to tune ATLAS based on local system configuration. The tuned atlas has advantage over the regular atlas with an almost 10-fold increase in speed. If Real-ESSI program is used for testing purpose only, one does not need to follow this (rather involved) step (of tuning of ATLAS), rely on the default installation of ATLAS and jump directly to section 209.6.5
209.6.4.1 [Optional Atlas Tuning :: Recommended for High Performance]

Real-ESSI is meant to take maximum advantage of your platform's hardware capabilities to provide a high-performance finite element implementation for real earthquake-soil-structure interaction simulations. An efficient implementation of BLAS (Basic Linear Algebra Subprograms) is crucial to attain this goal. Therefore, we have opted to use ATLAS (Automatically Tuned Linear Algebra Software) to provide a working, portable, high-performance BLAS and LAPACK as default for Real-ESSI.

A side-effect of this election is that end-users will have to go through the ATLAS auto-tuning process for their system in order to get performance out of Real-ESSI. If you want to skip this step, and provide your own version of BLAS and LAPACK, please see next section. The most import step for ATLAS auto-tuning process is

**Disabling Auto CPU Frequency Scaling**  
Auto scaling of CPU frequencies has to be turned OFF for ATLAS to be properly configured. Modern CPUs can scale their clock frequencies up and down, in order to respond to computational load and also save energy. This scaling needs to be turned off so that ATLAS can properly evaluate CPU performance, and tune its performance using all the features of a CPU.

This scaling is found on all current modern CPUs. So in order to turn off CPU frequency scaling you will need to install this utility:

```
1 sudo apt-get install cpufrequtils
```

Then edit the following file (if it doesn’t exist, create it):

```
1 sudo nano /etc/default/cpufrequtils
```

and add the following line to it:

```
1 GOVERNOR="performance"
```

then save and exit. You also need to disable ondemand daemon, otherwise after you reboot the settings will be overwritten.

```
1 sudo update-rc.d ondemand disable
```

Now you can reboot.

You will be able to restore the old ondemand daemon (which will control frequencies of your CPU(s), just like it did before, and have your CPU save energy when it is not needed), by doing
1  
```
sudo /etc/init.d/cpufrequtils restart
```

This will enable temporarily the "performance" governor, until next reboot.

After setup is done, check:

1  
```
cpufreq-info
```

and make sure that the current CPU-frequency for all CPUs is the hardware maximum. If not you can do:

1  
```
sudo /etc/init.d/cpufrequtils restart
```

Then re-check for the hardware maximum.

For more info see http://askubuntu.com/questions/523640/.

If you now run a command

1  
```
cpufreq-info
```

and you note that your CPU(s) (are) is at the maximum frequency already, you can probably skip steps below and jump to page 1299 where we continue to describe ATLAS build process.

**A Note on Intel CPUs and Linux Kernel \( \geq 3.9 \):** For new Intel processors running the Linux Kernel 3.9 or above the default driver for the CPU frequency scaling is `intel_pstate`. This has to be disabled via the Linux kernel command:

1  
```
intel_pstate=disable
```

so that the driver `acpi-cpufreq` takes over, allowing constant CPU frequency at the max frequency.

For example, on Ubuntu 14.04 one edits `/etc/default/grub` and change the line

1  
```
GRUB_CMDLINE_LINUX_DEFAULT="nomodeset quiet splash"
```

to

1  
```
GRUB_CMDLINE_LINUX_DEFAULT="nomodeset quiet splash intel_pstate=disable"
```

The new file `/etc/default/grub` should look something like this (this is example from one of our computers, your file will look similar but not the same, except for the addition of `intel_pstate=disable` to that line):

1  
```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
```
# info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="nomodeset quiet splash intel_pstate=disable"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command 'vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"

Then re-configure grub with

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

Reboot and confirm that this worked by running

```
cpufreq-info
```

and look at the driver information that should read acpi-cpufreq rather than intel_pstate.

**Separating "Real" CPUs from Hyperthreaded CPUs**  This is probably not of any interest for regular users, so please skip to the next section.

If you want to configure ATLAS to use multiple cores and hyperthreading\(^1\) than the above configure

\(^1\)THIS IS NOT TO BE USED FOR PARALLEL VERSION OF Real-ESSI AS YOU WILL BE EXPLICITLY USING THOSE MULTIPLE CORES YOURSELF!
options would look like this:

```bash
../configure -b 64 -D c -DPentiumCPS=2000 --prefix=$RealESSI_Dependencies_PATH
   --with-netlib-lapack-tarfile=$RealESSI_Dependencies_PATH/lapack-3.5.0.tgz
   --force-tids="4 0 1 2 3"
```

All the options, as described above are the same with an addition of

Explanation of options:

- `--force-tids="4 0 1 2 3"` tells ATLAS to only use those core whose ids are given (first number is the number of cores to use).

For nagoyqqatsi computer:

```bash
../configure -b 64 -D c -DPentiumCPS=2200 --prefix=$RealESSI_Dependencies_PATH
   --with-netlib-lapack-tarfile=$RealESSI_Dependencies_PATH/lapack-3.5.0.tgz
   --force-tids="8 0 1 2 3 4 5 6 7"
```

For an Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz:

```bash
../configure -b 64 -D c -DPentiumCPS=4000 --prefix=$RealESSI_Dependencies_PATH
   --with-netlib-lapack-tarfile=$RealESSI_Dependencies_PATH/lapack-3.5.0.tgz
   --force-tids="4 0 1 2 3"
```

Why is this important:

Modern processors provide hyperthreading [https://en.wikipedia.org/wiki/Hyper-threading](https://en.wikipedia.org/wiki/Hyper-threading) feature that creates virtual processor cores in an attempt to parallelize instruction execution when possible. This feature may or may not affect a particular platform, so some experimentation on part of the user is needed. If unsure, then just use only the amount of real cores available in your system.

Look at `cat /proc/cpuinfo` and look at the core ids. Pick processors which are on different cores.

On José’s laptop (quad core, Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz)

```bash
--force-tids="4 0 1 2 3"
```

On nagoyqqatsi computer (oct core, AMD Opteron(TM) Processor 6274 @ 2.20 GHz)

```bash
--force-tids="8 0 1 2 3 4 5 6 7"
```

It might help if you do:

```bash
cat /proc/cpuinfo | grep "core id"
```

It will show the number of real cores (and `cat /proc/cpuinfo` will show the processor ids associated with them).

Also READ the ATLAS manual under ”Handling hyperthreading, SMT, modules, and other horrors”.
Continuation of ATLAS tuning/build process  Continue tuning/building ATLAS

- Tune ATLAS to your system (example is for an Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz laptop, using the maximum turbo frequency of 4.40GHz)

For bash do:

1. `export RealESSI_Dependencies_PATH=set_to_appropriate_path`

or for tcsh do:

1. `set RealESSI_Dependencies_PATH=(set_to_appropriate_path)`

For example, in my case (Boris) this previous command looks like

1. `set RealESSI_Dependencies_PATH=('/home/jeremic/oofep/Rad_na_Sokocalu/ESSIforOTHERS/RealESSI_ROOT/RealESSI_Dependencies')`

check again your CPU frequency:

1. `cpufreq-info`

and provided that your CPU is at full throttle, start building ATLAS:

1. `mv ATLAS ATLAS3.10.x`
2. `cd ATLAS3.10.x`
3. `mkdir MY_CPU_type`
4. `cd MY_CPU_type`
5. `time ../configure -b 64 -D c -DPentiumCPS=2900 --prefix=$RealESSI_Dependencies_PATH/lib --with-netlib-lapack-tarfile=$RealESSI_Dependencies_PATH/lapack-3.5.0.tgz`

Explanation of options for configure:

- `-b` is the pointer bitwidth, 64 is standard
- `-D c -DPentiumCPS=2900` is a system-dependent setting, and it sets the CPU clock rate so that ATLAS can use CPU cycles for timing. You can get that measure running `cpufreq-info` and recording the highest possible frequency that your CPU supports. Unit for this argument is MHz (so for 2.20GHz you would write 2200)
- `--prefix` where to install, a good idea is `$RealESSI_Dependencies_PATH/lib`
- `--with-netlib-lapack-tarfile` where is the lapack tarball (provided in the Real-ESSI dependencies tarball), and that is where we have it in Real-ESSI dependencies and...
For nagoyqqatsi computer, this last line is

```bash
../configure -b 64 -D c -DPentiumCPS=2200 ←
  --prefix=$RealESSI_Dependencies_PATH ←
  --with-netlib-lapack-tarfile=$RealESSI_Dependencies_PATH/lapack-3.5.0.tgz
```

For an José's laptop, Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz this looks like:

```bash
../configure -b 64 -D c -DPentiumCPS=4000 ←
  --prefix=$RealESSI_Dependencies_PATH ←
  --with-netlib-lapack-tarfile=$RealESSI_Dependencies_PATH/lapack-3.5.0.tgz
```

For Boris' laptop:

```bash
time ../configure -b 64 -D c -DPentiumCPS=2900 ←
  --prefix=$RealESSI_Dependencies_PATH ←
  --with-netlib-lapack-tarfile=$RealESSI_Dependencies_PATH/lapack-3.5.0.tgz
```

You are now ready to build ATLAS. This process will take some time (between 7 and 30 minutes on our computers), do not run anything in parallel with this, let your computer devote full attention to ATLAS. In addition do not use parallel compile (as in `make -j 8` or similar, as this will skew ATLAS tuning. Proceed to build ATLAS:

```bash
time make
```

You can also perform some testing:

- sanity check correct answer:
  ```bash
  make check
  ```

- sanity check parallel
  ```bash
  make ptcheck
  ```

- check if lib is fast
  ```bash
  make time
  ```

Once this is done, install:

```bash
make install
```

This creates the 'lib' and 'include' directories in the RealESSI_Dependencies_PATH which are
expected by the Real-ESSI build system and needed for placing the rest of the libraries dependencies. Compilation will not continue without these.

209.6.5 Building Dependencies from Source

Now, we have all that we need to build dependencies. To build all the dependencies at once we need to run the `build_libraries` script. To compile the libraries for sequential or parallel version:

- **Build all Libraries**
  ```
  ./build_libraries NPROC=8
  ```

- **Build only sequential Libraries**
  ```
  ./build_libraries sequential NPROC=8
  ```

- **Build only parallel libraries Libraries**
  ```
  ./build_libraries parallel NPROC=8
  ```

  where argument `NPROC` is the number of processes, the user wants to run their makefile on. This is equivalent to `"-j"` option in makefile. By default, `NPROC=1`.

209.6.5.1 With Tuned ATLAS

On the top of that if the use wants to build a *TUNED ATLAS*, then one needs to add extra arguments to the build_libraries script as the following

- `TUNED_ATLAS=ON`
- `CPU_MAX_FREQ=max_cpu_frequency_of_the_system`

By default, Atlas tuning is off i.e. `TUNED_ATLAS=OFF`. With the addition of the above parameters (assuming the `max_cpu_freq` = 2000 MH) the commands would be

- **Build all Libraries**
  ```
  ./build_libraries TUNED_ATLAS=ON CPU_MAX_FREQ=2000 NPROC=20
  ```

- **Build only sequential Libraries**
1 ./build_libraries sequential TUNED_ATLAS=ON CPU_MAX_FREQ=2000 NPROC=20

- Build only parallel libraries Libraries

1 ./build_libraries parallel TUNED_ATLAS=ON CPU_MAX_FREQ=2000 NPROC=20

209.6.5.2 Checking the dependencies build libraries

The user can finally check all the libraries that are build by running the following arguments with `build_libraries` script

- Checking Sequential Libraries

1 ./build_libraries check_sequential

- Checking Parallel Libraries

1 ./build_libraries check_parallel

- Showing all libraries build

1 ./build_libraries list_build_dependencies

209.6.6 Compiling Real-ESSI Program Source

Real-ESSI sources are available to developers/collaborators under an open source license.

If you just want to run the program, you will need to have linux installed with appropriate libraries available (as detailed above). Installation of the Real-ESSI program is most optimal if remote users have available an up to date linux system (sequential and/or parallel) and if arrangements can be made for a temporary, simple/regular user, remote login (through a secure shell) for Prof. Jeremić. This will allow us to compile and install all the necessary libraries and the executable. Distribution of executable without remote login is available also, however in this case we distribute an un-optimized (slow), version of the Real-ESSI Program.

For developers with direct access to git repository do the following:

1 git pull
2 cd RealESSI/

then recompile:
209.6.7 Sequential Version

Compilation of the sequential version needs to be setup in Makefile. Compilers file where a line

```
PROGRAMMING_MODE = SEQUENTIAL
```

needs to be un-commented, while line `#PROGRAMMING_MODE = PARALLEL` needs to be commented out.

In addition to the above steps, compilation of the sequential version\(^2\) is done by executing:

```
mkdir build
cd build
time cmake ..
time make -j 16
```

Compilation is done in parallel using available CPUs/cores (here using 16 CPUs, command `time make -j 16`). Depending on the number of available cores/CPUs this number (16) can be changed to whatever is appropriate/desired/available. Number of CPUs will only change/improve the speed of compilation, while for building a parallel version of the Real-ESSI Program, different approach is used (to be described later). Main executable program for the Real-ESSI Simulator is in folder `build` and is named `essi`.

It is recommended to create a symbolic link from the Real-ESSI executable to a folder in the system’s path. For example:

```
sudo ln -sf $RealESSI_PATH/build/essi /usr/local/bin/essi
sudo chmod a+rx /usr/local/bin/essi
```

This will make Real-ESSI available to all users of the system.

Shell script `clean.sh` removes all the object files for all libraries, while leaving the main executable `essi` untouched.

\(^2\)Using 16 CPUs/Cores, if smaller/larger number is available/desired, please change that number in make command (time make -j 16).
209.6.8 Parallel Version

This section describes Real-ESSI build process on a distributed memory parallel machine. In addition to the above general packages, for parallel version the following packages need to be installed:

For Ubuntu 14.04:

1. `sudo apt-get install build-essential`
2. `sudo apt-get install cmake`
3. `sudo apt-get install openmpi-bin openmpi-doc libopenmpi-dev libopenmpi1.6`

Follow instructions for installing the dependencies. Use the script `compile_libraries_parallel.sh` after tuning ATLAS.

Compilation of parallel Real-ESSI needs execute the command in Real-ESSI source code folder.

1. `mkdir pbuild && cd pbuild`
2. `cmake -DCMAKE_CXX_COMPILER=/usr/bin/mpic++ -DPROGRAMMING_MODE=PARALLEL ..`
3. `make -j 32`

Once done, parallel Real-ESSI executable (named `essi`) is located in the build directory that resides within the source directory.

Again, it is a good idea to create a symbolic link from the Real-ESSI executable to a folder in the system’s path. For example:

1. `sudo ln -sf $RealESSI_PATH/pbuild/essi /usr/local/bin/essi_parallel`

This will make Real-ESSI available to all users of the system. Please note that we created a different name for a link `essi_parallel` to distinguish it from the sequential version (which can reside at the same time on the same system, input files will be exactly the same...).

209.7 Real-ESSI and OpenFOAM, Connecting

OpenFOAM is a free, open source computational fluid dynamics (CFD) software developed primarily by OpenCFD Ltd since 2004 (https://www.openfoam.com/). Real-ESSI supports numerical interface with OpenFOAM and can perform solid/structure fluid interaction analysis through Real-ESSI – OpenFOAM connection.

209.7.1 Installation of Customized OpenFOAM

We have made in-house modifications and developments to the InterFOAM application (Deshpande et al., 2012) of OpenFOAM-v1612+ for solid fluid interaction. This section presents the installation of the Customized OpenFOAM:
Install the dependencies:

```bash
1  sudo apt-get update
2  sudo apt-get install build-essential
3  sudo apt-get install flex
4  sudo apt-get install bison
5  sudo apt-get install cmake
6  sudo apt-get install zlib1g-dev
7  sudo apt-get install libboost-system-dev
8  sudo apt-get install libboost-thread-dev
9  sudo apt-get install libopenmpi-dev
10  sudo apt-get install openmpi-bin
11  sudo apt-get install gnuplot
12  sudo apt-get install libreadline-dev
13  sudo apt-get install libncurses-dev
14  sudo apt-get install libxt-dev
15  sudo apt-get install qt4-dev-tools
16  sudo apt-get install libqt4-dev
17  sudo apt-get install libqt4-opengl-dev
18  sudo apt-get install freeglut3-dev
19  sudo apt-get install libqtwebkit-dev
20  sudo apt-get install libscotch-dev
21  sudo apt-get install libcgal-dev
```

Also, make sure gcc and cmake meet the following minimum version requirements:

- gcc: version 4.8.5 or above
- cmake: version 3.3 or above

Check the version of gcc and cmake by running the following commands on terminal. If you are installing on Ubuntu 16.04 and above, the system version of gcc and cmake should already meet the requirements.

```bash
1  gcc --version
2  cmake --version
```

Downloaded the source code of Customized OpenFOAM:

```bash
1  wget http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/_Chapter_SoftwareHardware_Build_Process/OpenFOAM/sources/OpenFOAM.tar.gz
```

Choose a directory and extract the downloaded compressed file to the target directory.

```bash
1  tar -xzvf OpenFOAM.tar.gz -C /target/directory
```

For example, hereafter we choose $HOME as target directory. Replace $HOME with your chosen directory accordingly.

```bash
1  tar -xzvf OpenFOAM.tar.gz -C $HOME
```
Go to the extracted folder and source OpenFOAM environment configurations:

```
1 cd $HOME/OpenFOAM
2 source $HOME/OpenFOAM/OpenFOAM-v1612+/etc/bashrc
```

Setup CGAL and Boost version for compilation:

```
1 cgal_version=CGAL-4.9.1
2 boost_version=boost-system
```

Check the system readiness

```
1 foamSystemCheck
```

Change to the main OpenFOAM directory:

```
1 foam
```

**Note:** if running `foam` cannot change to the main OpenFOAM directory, in this case the directory is `$HOME/OpenFOAM`, source the environment configuration again by running the following terminal command.

```
1 source $HOME/OpenFOAM/OpenFOAM-v1612+/etc/bashrc
```

Compile OpenFOAM:

```
1 ./Allwmake
```

Since OpenFOAM is shipped with ParaView for post-processing OpenFOAM field results using developed plug-in paraFoam (https://cfd.direct/openfoam/user-guide/v6-paraview/). We also need to compile customized ParaView with paraFoam plug-in:

```
1 cd $WM_THIRD_PARTY_DIR
2 ./makeParaView
```

### 209.7.2 Check the Customized OpenFOAM Installation

Open a new terminal and source the OpenFOAM environment:

```
1 source $HOME/OpenFOAM/OpenFOAM-v1612+/etc/bashrc
```

Validate the build by running:

```
1 foamInstallationTest
```

Create a user run directory:
mkdir -p $FOAM_RUN

go to the user run directory:

run

Copy a simulation case from OpenFOAM tutorial to the user run directory:

cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily ./

go to the copies case directory:

cd pitzDaily

Generate the mesh:

blockMesh

Perform the analysis with the application simpleFoam:

simpleFoam

Visualize the simulation results:

paraFoam

209.7.3 Compile Real-ESSI with Link to OpenFOAM

Go to Real-ESSI source directory under directory RealESSI_ROOT and clean any previous old compilation of Real-ESSI:

cd RealESSI_ROOT/Real-ESSI
rm -rf bin
rm -rf lib
rm -rf build_sequential
mkdir bin
mkdir lib
mkdir build_sequential

cd build_sequential

Build and install the executable, using 16 CPUs in this case. Of course, if you have more CPUs available, you can use most of them. Please make sure to specify your OpenFOAM installation directory with CMake argument -DOPENFOAM_DIR. For example, in this case, we specify the installation directory as $HOME/OpenFOAM.
Rename essi to essi.sequential just so to distinguish it from the parallel executable:

```bash
cd ../bin
cp essi essi.sequential
```

Finally, install essi.sequential in system binary directory so that others can use it:

```bash
sudo rm /usr/bin/essi /usr/bin/essi.sequential
sudo cp essi.sequential /usr/bin/essi.sequential
sudo chmod a+x /usr/bin/essi.sequential
```

### 209.8 Code Verification After the Build Process

After build process, test cases to verify that installation is successful should be run. There are four groups of verification cases. The first two groups are designed for users. The last two groups are designed for developers.

1. The first group of test cases compares the sequential essi results to the analytic solutions.
2. The second group of test cases compares the parallel essi results to the analytic solutions.
3. The third group of test cases tests the version stability between two essi executables.
4. The fourth group of test cases tests the memory management of Real-ESSI with valgrind.

#### 209.8.1 Run all verification test cases

In order to run all test cases to verify the installation, users can run

```bash
cd $RealESSI_PATH/
bash run_all_verification.sh
```

Please make sure that sequential essi is available as "essi" in the PATH, and parallel essi is available as "essi_parallel" in the PATH before running all the verification test cases.

In addition, if users want to clean the test results, users can run

```bash
cd $RealESSI_PATH/
bash clean_all_verification.sh
```

Finally, users can also run a single group of test cases as follows.
209.8.2 Test Sequential Real-ESSI

In order to test whether the installation of sequential essi is successful, open the sequential example folder and run the bash script.

```bash
1 cd $RealESSI_PATH/CompGeoMechUCD_Miscellaneous/examples/analytic_solution
2 bash make_comparison.sh
```

This bash script will run all the examples automatically and compare the results to the analytic solutions. The comparison results are not only printed in the Terminal but also saved as a .log file in the same folder. Before you run the examples, make sure `essi` is in your PATH.

209.8.3 Test Parallel Real-ESSI

In order to test whether the installation of parallel essi is successful, open the parallel example folder and run the bash script.

```bash
1 cd $RealESSI_PATH/CompGeoMechUCD_Miscellaneous/examples/parallel
2 bash make_comparison.sh
```

This bash script will run all the examples automatically and compare the results to the analytic solutions. The comparison results are not only printed in the Terminal but also saved as a .log file in the same folder. Before you run the examples, make sure `essi_parallel` is in your PATH.

209.8.4 Version Stability Test

Since new features are continuously updated and improved in Real-ESSI, the version stability test helps the developers to guarantee their modification will not affect the correct operation of other code.

In order to test version stability,

```bash
1 cd $RealESSI_PATH/CompGeoMechUCD_Miscellaneous/examples/version_stability
2 bash generate_original.sh
```

This bash script will run all the examples automatically and save the results for reference later. This bash script above should run with the previous stable essi.

Then, to test the new essi and compare the results

```bash
1 cd $RealESSI_PATH/CompGeoMechUCD_Miscellaneous/examples/version_stability
2 bash make_comparison.sh
```

This bash script will run all the examples again and compare the results to the previous saved results. This bash script should run with the new essi. The comparison results are not only printed in the Terminal
but also saved as a .log file in the same folder.

### 209.8.5 Memory Management Test

Memory management is important in C/C++ programming. This group of test cases helps the developers to track the memory leak in Real-ESSI. For the details about the code stability verification, please refer to the Section 303.2 on Page 1396.

Before you run the test cases, make sure Valgrind is installed. You can install Valgrind by this command:

```bash
sudo apt-get install valgrind
```

You can also download the source of Valgrind and compile it from scratch.

It is important to test memory leak in parallel simulations.

```bash
mpirun -np 3 valgrind --log-file='log_%p.valgrind' --leak-check=yes essi-parallel-debug -f main.fei
```

A few important things to mention here:

- To test memory leak in parallel simulation, you obviously need a parallel version of Real-ESSI.
- Real-ESSI needs to be compiled in debug mode. This is important for Valgrind to capture and location the source of memory leaks.
- Running Real-ESSI in debug mode and in Valgrind means the simulation will be very slow. So it’s not practical to run memory leak test using a large model. You should have a model with only a few elements/nodes (but more than 1 element so that it runs in parallel) that includes the specific functions you want to test.
- Valgrind log files will be saved in the location where you run the model. There will be multiple log files named as `log_processID.valgrind`. Each process will have its own Valgrind log file. There might be a few empty Valgrind log files generated, you can just ignore those. The number of Valgrind log files that actually contain memory leak information should be the same as the number of cores you use in your simulation.
- Valgrind is a powerful tool with many options. The command shown above is rather basic but serves as a good starting point. Memory leaks can be very tricky to track and fix. You should learn and experiment with Valgrind options for different issues you want to fix.
Valgrind log file can be very long and hard to read. At the bottom, there is a leak summary that looks like this: You should primarily focus on the 'definitely lost' result. ‘Indirectly lost’ and ‘possibly lost’ can also be problematic but should go away once you fix the source of ‘definitely lost’. ‘Still reachable’ is usually not considered as actual memory leak but is something that can be optimized. Refer to the Valgrind User Manual for more information.

Valgrind log file contains detailed information on each memory leak. A typical leak detail looks like this. Following the trail, you should be able to locate the source of a specific leak and then fix it properly.

A serious memory leak issue caused by external solvers used by PETSc was found. As shown in Figure 209.3, when the \texttt{mumps} option was used in parallel solver, a significant amount of memory leak was detected by Valgrind. More importantly, such memory leak was observed to increase with the number of time steps. This means large-scale, long-duration simulation could be interrupted due to not enough memory in the operating system. Note that this issue was also reported in other occasions where the \texttt{mumps} package is used within PETSc, as recent as June 2020.

After extensive tests, it has been found out that other options/packages in PETSc don’t have the memory leak issue mentioned above. Therefore it is recommended to use options other than \texttt{mumps} for large-scale, long-duration simulations. For example, the following command calls the default direct solver of PETSc:

```bash
define solver parallel petsc "-ksp_type preonly -pc_type lu" ;
```
Real-ESSI Lecture Notes  209.9.  COMPILING REAL-ESSI UTILITIES

209.9  Compiling Real-ESSI Utilities

Real-ESSI comes with a lot of utilities to help the users speed up the simulation process. It provides mesh building, auto-input generation and visualization features which makes it quite nice.

Real-ESSI source code contains build_utilities script which can be used to build all the available utilities. We will go through the following subsection to introduce each utility and how to compile them.

The first step is to download all the sources of utilities that needs to be build. To do this, one has to run

```
1 cd Real-ESSI
2 ./build_utilities download
```

This would download all the utilities sources in tar.gz format and would place them in "/SRC" of RealESSI_Utilities directory. The script is very powerful and accepts targets that can be used to build a particular utility or all utilities at once. The available options to the scripts can be found by running the target help as shown below. A snippet is shown below

```
1 ./build_utilities help
2 #usage: make [target]
3 #
4 #Utilities:
5 # gmessi Builds gmessi
6 # paraview Builds paraview
7 # pvessi Builds pvessi
8 # gmsh Builds gmsh
```
The user can compile individual utilities by running just running

```
./build_utilities <utility_name>
```

**Note:** All the binaries of the utilities after build gets linked/copied to the `RealESSI_Utilsities/bin` directory inside `RealESSI_ROOT`.

### 209.9.1 Installation of gmsh and gmESSI

**gmsh** is a 3-D finite element mesh generator for academic problems with parametric input and advanced visualization capabilities. It can be downloaded and installed from [http://geuz.org/gmsh/](http://geuz.org/gmsh/). Additionally, the user can also install gmsh from terminal:
sudo apt-get install gmsh

**gmESSI** is effective pre-processor for generating Real-ESSI input files directly for the mesh file provided by gmsh. More information about gmESSI and how it works is given in Chapter 207 of the main document, lecture notes (Jeremić et al., 1989-2021). The gmESSI package is available from the main repository site: [http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/_Chapter_SoftwareHardware_PreProcessing/Real-ESSI_gmESSI.tgz](http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/_Chapter_SoftwareHardware_PreProcessing/Real-ESSI_gmESSI.tgz). Before installing gmESSI, please install required libraries, as explained in Section 209.5.2.2 on Page 1284. To install gmESSI, go to the Real-ESSI directory and then run

1. ./build_utilities gmessi

To update the utility at the user just needs to run

1. ./build_utilities update_gmessi

Refer to section 209.10 on page 1319 for instructions on what and how to install autocompletion and syntax coloring for gmESSI and Real-ESSI syntax on sublime text editor.

## 209.9.2 Installation of ParaView and PVESSIReader

**ParaView** package [http://www.paraview.org/](http://www.paraview.org/) is a powerful multi-platform data analysis and visualization application available in Open Source. It can be run on supercomputers to analyze datasets of petascale size as well as on laptops for small datasets. ParaView can be used to visualize results of Real-ESSI simulations. A plug-in was developed for ParaView so that all the simulations results from Real-ESSI finite elements, material models and analysis types can be directly visualized, animated, etc.

### 209.9.2.1 Building ParaView and PVESSIReader Plugin from Source on Linux System

**Building ParaView**

1. Install dependencies

```
1  sudo apt-get install libphonon-dev libphonon4 qt4-dev-tools
2  sudo apt-get install libqt4-core libqt4-gui qt4-qmake libxt-dev
3  sudo apt-get install g++ gcc cmake-curses-gui libqt4-opengl-dev
4  sudo apt-get install mesa-common-dev python-dev
5  sudo apt-get install librvtk6.2
```

2. After successful installation of dependencies, download the source files of Paraview. To ensure version compatibility, Paraview v5.2.0 can be downloaded from local server. Create and to the
directory where you want to put the Real-ESSI utilities and follow the commands shown below. Make sure the directory is clean before the installation.

```
1  mkdir ParaView
2  cd ParaView
3  
4  # using curly brackets to help in checking scripts, that rely on these
5  # brackets being available around URL
6  
7  wget {http://sokocalo.engr.ucdavis.edu/~jeremic/RealESSI/Utilities_SRC.tar.gz}
8  tar -xzvf ./Utilities_SRC.tar.gz Utilities_SRC/ParaView-v5.2.0-RC1.tar.gz
9  rm -rf Utilities_SRC.tar.gz
10  mkdir ParaView_INSTALL
11  tar -xzvf Utilities_SRC/ParaView-v5.2.0-RC1.tar.gz -C ParaView_INSTALL --strip-components 1
12  rm -rf Utilities_SRC
```

3. Compile ParaView:

```
1  cd ParaView_INSTALL
2  mkdir build
3  cd build
4  cmake .. -DPARAVIEW_USE_MPI=ON -DPARAVIEW_ENABLE_PYTHON=ON
5  make -j 4
```

4. ParaView executable is in /Paraview/build/bin. It is recommended to create a symbolic link from the ParaView executable to a folder in system path. You need to be an administrator to do so.

```
1  cd bin
2  sudo ln -sf ~/FULL_INSTALL_DIRECTORY/paraview /usr/local/bin/
3  # for example
4  sudo ln -sf ~/oofep/Rad_na_Sokocalu/ParaView/ParaView_INSTALL/build/bin/paraview /
   /usr/local/bin/
```

**Building PVESSIRReader Plugin**

1. By placing source files of PVESSIRReader in Plugins directory, PVESSIRReader will be automatically compiled together with ParaView. Start by going to ParaView Plugins directory

```
1  cd ../../Plugins
```

2. Obtain the latest version of the PVESSIRReader plugin:

```
1  #
```
3. Recompile part of ParaView with the PVESSIReader plugin present:

```bash
# using curly brackets to help in checking scripts, that rely on these brackets being available around URL
wget
    {http://sokocalo.engr.ucdavis.edu/~jeremic/RealESSI/Utilities_SRC.tar.gz}
tar -xzvf Utilities_SRC.tar.gz
tar -xzvf Utilities_SRC/PVESSIReader-master.tar.gz
rm -rf Utilities_SRC.tar.gz
%tar -xzvf Utilities_SRC/PVESSIReader-master.tar.gz
rm -rf Utilities_SRC
```

4. ParaView is now compiled with PVESSIReader.

5. Load PVESSIReader plugin into ParaView:

   - open ParaView
   - Click on Tools → Manage Plugins (See Figure 209.4)

   ![Figure 209.4: Open ParaView plugin-manager.](image)

   - Find PVESSIReader and click "Load Selected", see Figure 209.5,
• Click on PVESSIReader and check on the "Auto Load" option, see Figure 209.5,

![ParaView Plugin Manager](image)

Figure 209.5: ParaView Plugin Manager.

• Close ParaView

• Start Paraview again. PVESSIReader is now part of automatically loaded plugins for ParaView and Real-ESSI results can be loaded by loading Real-ESSI HDF5 outp files that are named after model and stage name and have extension .h5.feioutput.

209.9.2.2 Building ParaView and PVESSIReader Plugin from Source on Windows System

Note that ParaView, as well as its building procedure, has recently (during 2020) gone through some major changes. The building procedures shown in this section are mostly based on the information available at: [https://gitlab.kitware.com/paraview/paraview/blob/master/Documentation/dev/build.md](https://gitlab.kitware.com/paraview/paraview/blob/master/Documentation/dev/build.md). It is noted that user should be prepared to spend some time (perhaps hours) on installing procedure...

1. Install Dependencies

• Download and install git bash for windows. Use the latest release version.

• Download and install cmake. Use the latest release version.

• Download and install Visual Studio 2015 Community Edition. If this is the first time Visual Studio is installed in your system, some developer tools are probably missing. Open Visual Studio and attempt to build a new Visual C++ project. If you see any options saying "Install ... for C++", click on that/them and necessary tools will be installed.

• Download ninja-build and drop ninja.exe in C:\Windows\ Use the latest release version.
Download and install both msmpisetup.exe and msmpisdk.msi from Microsoft MPI. Use the latest release version from Microsoft.

Download and install Python for Windows. Latest release version should work fine. To avoid potential compatibility issues, install the same Python version that is used for the latest release of ParaView.

Download and install Qt 5.12.3 for Windows, make sure to check the MSVC 2015 64-bit component during installation, make sure to add C:\Qt\Qt5.12.3\5.12.3\msvc2015_64\bin to your PATH environment variable. Note that Qt for Windows is x86 but it works for x64 machine as well.

2. Obtain the source of ParaView

Open your preferred Windows command prompt. Windows PowerShell is a nice tool for people usually work with Linux system.

To build ParaView development version (usually referred as "master"), run the following commands:

```
1 cd C:
2 mkdir pv
3 cd pv
4 git clone --recursive https://gitlab.kitware.com/paraview/paraview.git
5 mv paraview pv
6 mkdir pvb
```

To build a specific ParaView version, please refer to https://gitlab.kitware.com/paraview/paraview/blob/master/Documentation/dev/build.md.

3. Obtain the source of PVESSIReader plugin

The source of PVESSIReader plugin can be downloaded from http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/_Chapter_SoftwareHardware_Post_Processing/_Real_ESSI_PVESSIReader_.zip

Extract the files and move the PVESSIReader folder to C:\pv\Plugins\n
4. Modify the cmake file to include PVESSIReader plugin in the building process. Open the file C:\pv\CMakeLists.txt using your choice of text editor. Find "set(paraview_default_plugins" and add "PVESSIReader" to the end of the list of plugins.

5. Build
Open VS2015 x64 Native Tools Command Prompt and run the following commands:

```
1  cd C:\pv\pvb
2  cmake -GNinja -DPARAVIEW_USE_PYTHON=ON -DPARAVIEW_USE_MPI=ON -DCMAKE_BUILD_TYPE=Release ..\pv
3  ninja
```

This step could take a few hours. If no configuration or compilation error is encountered, you should have the ParaView executable at `C:\pv\pvb\bin\`.

6. Load PVESSEISReader plugin into ParaView

- Run the ParaView executable and click on Tools → Manage Plugins → Load New ...
- Find `PVESSEISReader.dll` at `C:\pv\pvb\bin\paraview-5.8\plugins\PVESSEISReader\` and click OK to load it.
- Now you should see PVESSEISReader loaded in the list of plugins. Double click on it to expand advanced options and check Auto Load.
- Close the ParaView application and reopen it. Now the PVESSEISReader plugin should be automatically loaded and ready to use.

### 209.10 Sublime Text Editor

Install sublime text editor from [http://www.sublimetext.com/](http://www.sublimetext.com/). Then install package control to sublime in order to install plugins. (go to preferences, package control, install package.) Then install two packages:

- **FEI Syntax-n-Snippets**, Real-ESSI syntax and auto completion plugin for `.fei` files (input files for Real-ESSI program).
- **gmsh-Tools**, syntax and autotext completion for gmsh model development tools for Real-ESSI.
- **gmESSI-Tools**, syntax and autotext completion for gmESSI model development tools for Real-ESSI.

### 209.11 Model Conversion/Translation using FeConv

FeConv allows conversion/translation of input files (models) between Real-ESSI and SASSI, Sofistik, Ansys, OpenSees and Strudyn. FeConv was developed and is maintained by Mr. Viktor Vlaski.
209.12 Build Procedures on Amazon Web Service

This section shows the steps to install a new Real-ESSI image on Amazon Web Service (AWS). This document is only intended for Real-ESSI developers, not for general users. For using Real-ESSI on AWS, please refer to Chapter 211, on page 1338 in Jeremić et al. (1989-2021).

Noted that when creating a new image, the instance type should be consistent with future usage. For example, if the user intend to launch a Real-ESSI instance using the instance type "General Purpose", such as the T2 series, the image should also be created with the same instance type. If the image is created with a different instance type, Real-ESSI will not be able to run, and the following error message will be observed:

```
Illegal instruction (core dumped)
```

209.12.1 Sign In to AWS

Here is the link to the AWS sign in page. Click "Sign In to the Console" button on the upper right corner of the page. No need to register a new account. You should already have the account ID, IAM user name, and password for AWS sign in. If not, please contact an administrator to add you to the developers’ group.

After sign in, go to the "EC2" tab under "Service". Here you can view all your instances and AMIs. This is where you can start new simulations or install new images.

Note that you probably also need to choose the correct region. On the upper right corner of the page, you can see your current region and switch to another one if necessary.

209.12.2 Copy an Existing Image

Since we already have a few images for Real-ESSI, the most efficient way to create a new image is to simply copy an existing one. To do this, go to the "AMIs" tab under "IMAGES" on the left part of the page. Now you should be able to view all existing images.

Select the image that you want to copy. Click the "Actions" button and choose "Copy AMI". On the pop-up window, enter the informations of this new image that you want to create. Then just click the "Copy AMI" button.

Now you should have a new image that has been installed with all the Real-ESSI components. To make any change inside this image, you need to launch it as a new instance and access it using X2GO. Procedures to install and use X2GO can be found in Chapter 211. For cloud server, on AWS or similar, the build procedures are the same as those for local installation, which can be found in previous sections.
of this chapter.

### 209.12.3 Create a New Image

If you need to create a new Real-ESSI image from scratch, this section shows the steps to do so. First sign in to AWS and go to "EC2". Choose the correct region. Click the "Instance" tab under "INSTANCES" on the left part of the page. Choose "Launch Instance" to start a new instance that later will be saved as your new image.

Then, follow these steps:

2. Choose Instance Type: Family = Compute optimized, Type = c5.4xlarge, vCPUs = 16, Memory (GiB) = 32.
3. Keep other options as default, and click "Review and Launch".
4. Review the information of the new instance, and click "Launch".

Next, you are asked to choose a key pair for your instance. It’s recommended to create a new key pair for the first time, then use it in the future. First, choose "Create a new key pair", and enter a name. Click the "Download Key Pair" button. Save the key in a secure directory in your local computer for future use, for example in .ssh directory.

Now, you can select "Choose an existing key pair", and select your key pair that should be visible. Check the box for acknowledging the use of a private key. Finally, your new instance is launched. Note that this new instance is a brand new Ubuntu server, which means that you need to install everything.

At this point, the new Ubuntu server on AWS does not have X2GO for remote access or a GUI desktop to operate. We will now install these necessary softwares. First, run the following command to access the remote Ubuntu server on AWS using ssh. Note that you need to change the name of your ssh key to the one you just created. The public IP address can be found on the AWS webpage where you launched your new instance. Go the description of your instance to find the "IPv4 Public IP".

```
1 chmod 400 your_ssh_key.pem
2 ssh -i your_ssh_key.pem ubuntu@your_AWS_public_IP_address
```

Run the following command to install X2GO server on Ubuntu Linux.

```
1 sudo apt-get install software-properties-common
2 sudo add-apt-repository ppa:x2go/stable
3 sudo apt-get update
4 sudo apt-get install x2goserver x2goserver-xsession
```
Xfce is a lightweight desktop and ideal for usage on a remote server. Run the following command to install xfce on Ubuntu.

```
1  sudo apt-get install xfce4 xfce4-goodies
```

Now you can access your new instance (the remote Ubuntu server) using X2GO. Steps to do this can be found in Chapter 211. After you established remote control of the Ubuntu server on AWS, the build procedures are the same as those for local installation, which can be found in previous sections of this chapter.

The last step is to create a new image from this instance so that you can launch it in the future. Go the "Instances", and choose the correct instance. Click "Actions", and select "Create Image" under "Image". You can change the size of the instance volume, but it’s not necessary at this moment. Give your image a name and a description, and click "Create Image". Now you have sucessfully created a new image for Real-ESSI. If you go to "AMIs", you should be able to see this new image you just created.
209.12.4 Build AWS ESSI Image from Scratch

This section is a developer guide, which presents the procedures to build AWS ESSI Image from scratch. ESSI AWS users do not need to know the technical details in this section.


   EC2 Dashboard → Instances → Instance → Launch Instance.

   Choose

   | Ubuntu Server 16.04 (HVM), SSD Volume Type. |

   Since there is no Desktop version available, so we have to launch the server version and install desktop by ourself.

   You will need to download a .pem key to launch the instance.

2. Login to the Remote Instance using Terminal.

   Copy the external IP address of the remote instance from the Browser.

   Use the downloaded .pem key to login to the remote instance.

   | chmod 400 your_key.pem  
   | ssh -i your_key.pem ubuntu@your_remote_instance_IP |

3. Install Desktop and git on AWS Remote Instance

   | sudo apt update  
   | sudo apt install -y ubuntu-desktop git |

4. Install remote-desktop-server (x2goserver) on AWS Remote Instance

   | sudo add-apt-repository ppa:x2go/stable  
   | sudo apt update  
   | sudo apt install -y x2goserver x2goserver-xsession xfce4 |

5. Set up the automatic launch of remote desktop server

   | sudo systemctl enable x2goserver.service  
   | sudo systemctl start x2goserver.service |

6. Install ESSI
# Install prerequisite
sudo apt install -y cmake
sudo apt install -y build-essential
sudo apt install -y zlib1g-dev
sudo apt install -y libtbb-dev
sudo apt install -y bison flex
sudo apt install -y libboost-dev
sudo apt install -y python
sudo apt install -y gfortran
sudo apt install -y libopenblas-dev
sudo apt install -y liblapack-dev
sudo apt install -y python-scipy
sudo apt install -y libhdf5-dev libhdf5-cpp-11
sudo apt install -y python-h5py
sudo apt install -y python-matplotlib
sudo apt install -y libssl-dev

# Download ESSI
# using curly brackets to help in checking scripts, that rely on these
# brackets being available around URL
#
git clone {https://github.com/BorisJeremic/Real-ESSI.git} # Need ←
  permission from Boris Jeremic for Real-ESSI on github
cd Real-ESSI

# Build ESSI Dependencies
./build_libraries download
./build_libraries sequential
./build_libraries hdf5_sequential
./build_libraries suitesparse
./build_libraries arpack
./build_libraries parmetis
./build_libraries petsc

# Build Sequential ESSI
mkdir build
cd build
cmake ..
make -j $(nproc)
cd ..

# Build Parallel ESSI
mkdir build_parallel
cd build_parallel
cmake -DCMAKE_CXX_COMPILER=/usr/bin/mpic++ -DPETSC_HAS_MUMPS=TRUE ←
  -DPROGRAMMING_MODE=PARALLEL ..
make -j $(nproc)
cd ..
7. Install gmsh

```bash
sudo apt install -y gmsh
```

8. Install gmESSI

```bash
# Install the prerequisite
sudo apt install -y libboost-all-dev
sudo apt install -y build-essential
sudo apt install -y python-dev
sudo apt install -y liboctave-dev

# Install gmESSI
## download the package from the main Real-ESSI repository
# # using curly brackets to help in checking scripts, that rely on these
# # brackets being available around URL
#
wget
mkdir Real-ESSI-gmESSI
mv _all_files_gmESSI_.tgz Real-ESSI-gmESSI
cd Real-ESSI-gmESSI
make -j $(nproc)

# Add binary PATH to ~/.bashrc
cd ./build/bin/
part1="export PATH="
part2=$PWD
part3=":\$PATH"
newline=$part1$part2$part3
echo $newline >> ~/.bashrc
```

9. Install ParaView with PVESSIReader plugin

```bash
# Install the prerequisite
sudo apt install -y libavformat-dev
sudo apt install -y libswscale-dev
sudo apt install -y ffmpeg
sudo apt install -y libphonon-dev libphonon4 qt4-dev-tools
sudo apt install -y libqt4-core libqt4-gui qt4-qmake libxt-dev
sudo apt install -y g++ gcc cmake-curses-gui libqt4-opengl-dev
sudo apt install -y mesa-common-dev python-dev
sudo apt install -y libvtk6.2
sudo apt install -y mpich libopenmpi-dev
sudo apt install -y libxml-dev libxi-dev

# Download the ParaView
#```
# using curly brackets to help in checking scripts, that rely on these
# brackets being available around URL
#
git clone {https://github.com/Kitware/ParaView.git}
cd ParaView
git checkout v5.1.2
git submodule update --init

# Download the Plugin
cd Plugins
#
# using curly brackets to help in checking scripts, that rely on these
# brackets being available around URL
#
grep ←
  {http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/pvESSI/_pvESSI_all_files_.tgz}
tar -xvzf _pvESSI_all_files_.tgz
cd ..

# Compile ParaView along with PVESSIReader
mkdir build & cd build
cmake -DPARAVIEW_USE_MPI=true -DPARAVIEW_ENABLE_PYTHON=true ←
  -DPARAVIEW_ENABLE_FFMPEG=true ..
make -j $(nproc) # require Internet during ParaView compilation.

# Add binary PATH to "~/.bashrc
cd bin
part1="export PATH="
part2=$PWD
part3=":\$PATH"
newline=$part1$part2$part3
echo $newline >> ~/.bashrc

10. Install Sublime Text 3 and ESSI plugin. Following this link.

# using curly brackets to help in checking scripts, that rely on these
# brackets being available around URL
#
grep -qO - {https://download.sublimetext.com/sublimehq-pub.gpg} | sudo ←
  apt-key add -
sudo apt-get install apt-transport-https
echo "deb {https://download.sublimetext.com/ apt/stable/}" | sudo tee ←
  /etc/apt/sources.list.d/sublime-text.list
sudo apt-get update
dsud apt-get install sublime-text

11. Install Sublime Text Plugin:

# Inside Sublime Text Window
12. Create Image inside Browser.

Select the launched Image with the above software installed.

Choose Actions → Image → Create Image.

Type your Image Name and descriptions.

You will then see your image in EC2 Dashboard → Images → AMIs

### 209.12.5 Update an Existing Image

For updating an existing image, for example for a new version or release follow instruction below. First sign in to AWS and go to "EC2". Choose the correct region. Click the "Instance" tab under "INSTANCES" on the left part of the page. Choose "Launch Instance" to start a new instance that later will be saved as your new image.

Then, follow these steps:

1. Choose an existing AMI, for example GlobalRelease...

2. Choose Instance Type, for example: Family = Compute optimized, Type = c5.4xlarge, vCPUs = 16, Memory (GiB) = 32.

3. Keep other options as default, and click "Review and Launch".

4. Review the information of the new instance, and click "Launch".

Next, you are asked to choose a key pair for your instance. It’s recommended to create a new key pair for the first time, then use it in the future. That is the keypair that is saved, for example in .ssh.
Now, you can select "Choose an existing key pair", and select your key pair that should be visible. Check the box for acknowledging the use of a private key. Finally, your new instance is launched. Note that this new instance is an already existing Ubuntu server/image. This image is the one we will update.

Now you can access your new instance (the remote Ubuntu server) using X2GO. Steps to do this can be found in Chapter 211. After you established remote control of the Ubuntu server on AWS, the build procedures are the same as those for local installation, which can be found in previous sections of this chapter.

The last step is to create a new image from this instance so that you can launch it in the future. Go the "Instances", and choose the correct instance. Click "Actions", and select "Create Image" under "Image". You can change the size of the instance volume, but it’s not necessary at this moment. Give your image a (new) name and a description, and click "Create Image". Now you have successfully created a new image for Real-ESSI. If you go to "AMIs", you should be able to see this new image you just created.

Now you can go to Software directory and follow install procedures from section 209.5 on page 1282.

After compiling and linking both sequential and parallel Real-ESSI, and install them on /usr/bin (follow procedures for build), and delete source code (!), one can make this instance into a new image. Create new image inside AWS EC2 Management Console Browser window. Select the launched Image with the above software installed. Choose Actions → Image → Create Image. Type your Image Name and descriptions. Click Create Image. This might take some time. You will then see your image in EC2 Dashboard → Images → AMIs (on the left side bar).

Make sure that you terminate all the running instances so that you do not get charged. Find: Action, Instance State, Terminate.

209.12.6 Upload an Existing Real-ESSI Simulator Image to AWS Marketplace

- Copy to private image for region North Virginia
- Go to the AWS market place https://aws.amazon.com/marketplace,
- Choose sell in AWS marketplace,
- Choose AMIs selection the new private in Region North Virginia to publish.
- Proceed until finalizing the AWS Marketplace Image.
Chapter 210

Software Platform Procurement, Distribution

(In collaboration with Dr. Han Yang and Dr. Hexiang Wang)
210.1 Chapter Summary and Highlights

210.2 Real-ESSI Program Executables Download and Install

Executables for the Real-ESSI Simulator program (Jeremić et al., 1988-2021) are available online. Pre-built executables are available for Linux, Ubuntu 18.04, and can be downloaded and installed by analyst.

In order for prebuild executables to be able to run on a user/analyst computer, system libraries have to be brought up to date and additional libraries installed. System libraries update/upgrade:

```bash
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
sudo apt-get autoremove
```

For sequential and/or parallel version of Real-ESSI, additional libraries are needed, as described below.

210.2.1 Sequential Version of Real-ESSI Program.

Libraries required to be installed for using sequential version of the Real ESSI program:

```bash
sudo apt-get install libboost-all-dev
sudo apt-get install libhdf5-dev
sudo apt-get install libtbb-dev
sudo apt-get install libssl1.0.0
```

210.2.2 Parallel Version of Real-ESSI Program.

Libraries required to be installed for executing parallel version of the Real ESSI program:

```bash
sudo apt-get install libboost-all-dev
sudo apt-get install libhdf5-dev
sudo apt-get install libtbb-dev
sudo apt-get install mpich
sudo apt-get install libopenmpi-dev
sudo apt-get install libssl1.0.0
```

210.2.3 Real-ESSI Executable Downloads.

The Real-ESSI program executables can be downloaded from Real-ESSI Simulator web site: [http://real-essi.info/](http://real-essi.info/). Alternatively, contact Prof. Jeremić to arrange for customized Real-ESSI executables.
210.3 Real-ESSI Simulator Install as Container through Docker

Recent developments in virtualization of operating systems (OS) has created an opportunity to deploy programs and software systems as container images. Container images are used by the host OS (Linux, Windows, MacOS) to create a container. A container is a running instance of a container image, and is represented by a Linux/Windows/MacOS process that can be used to run programs that are installed within container. Programs that are installed within a container have all the necessary libraries available within container and are fully self sufficient, irrespective of what container host OS is used, be it Linux or Windows or MacOS.

More information used virtualization, containers, docker, etc. can be found at:

- [https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/](https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/)

Starting from Real-ESSI version 20.07, Real-ESSI Simulator is now available as a Docker Container Image, and can be installed and used on Linux, Windows and MacOS.

210.3.1 Real-ESSI Docker Image Development

This section is intended for Real-ESSI developers, users can skip this section. The development of Real-ESSI Docker image follows typical steps to ‘dockerize’ any application. Here are some very helpful sources:

- Official documentation: [https://docs.docker.com/](https://docs.docker.com/)
- A Docker Tutorial for Beginners: [https://docker-curriculum.com/#our-first-image](https://docker-curriculum.com/#our-first-image)
- How to dockerize any application: [https://hackernoon.com/how-to-dockerize-any-application-b60ad00e76da](https://hackernoon.com/how-to-dockerize-any-application-b60ad00e76da)
- Slimming Down Your Docker Images: [https://towardsdatascience.com/slimming-down-your-docker-image](https://towardsdatascience.com/slimming-down-your-docker-image)

It should be mentioned that there are many different ways and styles that can be employed to create Docker image. Here, multistage build is used to save build/debug time and, more importantly, reduce size of the final image.

Provided below are steps used to create the Real-ESSI Docker image.

- Obtain the source code of Real-ESSI.
• The following `Dockerfile` is created to build the Real-ESSI Docker image.

```bash
FROM ubuntu:18.04 AS basesystem

MAINTAINER Han Yang <hhhyang@ucdavis.edu>

WORKDIR /usr/src

COPY . .

RUN useradd -m ubuntu && \\
    apt-get update && apt-get install -y \\
    bison \\
    build-essential \\
    cmake \\
    flex \\
    libboost-all-dev \\
    libhdf5-serial-dev \\
    liblapack-dev \\
    libopenblas-dev \\
    libopenmpi-dev \\
    libpthread-workqueue-dev \\
    libssl-dev \\
    libtbb-dev \\
    mpich \\
    ssh \\
    valgrind \\
    wget \\
    zlib1g-dev

FROM basesystem AS dependencies

RUN cd Real-ESSI && \\
    mkdir -p ../RealESSI_Dependencies && \\
    mkdir -p ../RealESSI_Dependencies/include && \\
    mkdir -p ../RealESSI_Dependencies/lib && \\
    mkdir -p ../RealESSI_Dependencies/bin && \\
    mkdir -p ../RealESSI_Dependencies/SRC && \\
    cd ../Real-ESSI && \\
    ./build_libraries suitesparse && \\
    ./build_libraries arpack && \\
    ./build_libraries hdf5_sequential && \\
    ./build_libraries tbb && \\
    ./build_libraries lapack && \\
```

```
FROM basesystem AS dependencies

RUN cd Real-ESSI && \\
    mkdir -p ../RealESSI_Dependencies && \\
    mkdir -p ../RealESSI_Dependencies/include && \\
    mkdir -p ../RealESSI_Dependencies/lib && \\
    mkdir -p ../RealESSI_Dependencies/bin && \\
    mkdir -p ../RealESSI_Dependencies/SRC && \\
    cd ../Real-ESSI && \\
    ./build_libraries suitesparse && \\
    ./build_libraries arpack && \\
    ./build_libraries hdf5_sequential && \\
    ./build_libraries tbb && \\
    ./build_libraries lapack && \\
```

```
FROM basesystem AS dependencies

RUN cd Real-ESSI && \\
    mkdir -p ../RealESSI_Dependencies && \\
    mkdir -p ../RealESSI_Dependencies/include && \\
    mkdir -p ../RealESSI_Dependencies/lib && \\
    mkdir -p ../RealESSI_Dependencies/bin && \\
    mkdir -p ../RealESSI_Dependencies/SRC && \\
    cd ../Real-ESSI && \\
    ./build_libraries suitesparse && \\
    ./build_libraries arpack && \\
    ./build_libraries hdf5_sequential && \\
    ./build_libraries tbb && \\
    ./build_libraries lapack && \\
```
FROM dependencies AS builder

RUN cd Real-ESSI && \
    mkdir build && \
    cd build && \
    cmake .. && \
    make -j 16 && \
    cp essi essi_sequential && \
    cd .. && \
    mkdir pbuild && \
    cd pbuild && \
    cmake -DCMAKE_CXX_COMPILER=/usr/bin/mpic++ -DPROGRAMMING_MODE=PARALLEL .. && \
    make -j 16 && \
    cp essi essi_parallel

FROM ubuntu:18.04

MAINTAINER Han Yang <hhhyang@ucdavis.edu>

RUN useradd -m ubuntu && \
    apt-get update && apt-get install -y \
    libboost-all-dev \
    libhdf5-dev \
    libopenmpi-dev \
    libtbb-dev \
    mpich \
    ssh

COPY --from=builder /usr/src/Real-ESSI/build/essi_sequential \
     /usr/src/Real-ESSI/pbuild/essi_parallel /usr/bin/

USER ubuntu

WORKDIR /workspace

VOLUME ["/workspace"]

• Put the 'Dockerfile' in the same directory with the source code of Real-ESSI.

• Build the Real-ESSI Docker image. This step usually takes a long time, especially for the first time.
```bash
docker build -t realessilocal:test .
```

- Correctly tag your image. This is not only necessary for later push but also just a good practice to organize your Docker images.

```bash
docker tag realessilocal:test realessi/real-essi-repo:<tag>
```

Replace `<tag>` with the tag you want to use. It's usually a version name.

- Push your build to Docker Hub. Make sure you have the proper permission to do so.

```bash
docker push realessi/real-essi-repo:<tag>
```

### 210.3.2 Running Real-ESSI Container through Docker

Provided below are steps needed to install and run Real-ESSI within a Docker Container. The following steps work for both Linux and Windows systems. In a Linux system, run the following commands in a terminal. In a Windows system, run these commands in PowerShell. It should also work for Mac OS but hasn’t been tested yet.

- Install Docker on the local computer, desktop, laptop. Documentation on how to install Docker on user OS can be found here:

  - Linux: [https://docs.docker.com/engine/install/#server](https://docs.docker.com/engine/install/#server)
  - Windows: [https://docs.docker.com/docker-for-windows/install/](https://docs.docker.com/docker-for-windows/install/)
  - MacOS: [https://docs.docker.com/docker-for-mac/install/](https://docs.docker.com/docker-for-mac/install/)

- Manage Docker as a non-root user on Linux hosts

If you are using a Linux host, by default you need to run Docker using `sudo`. If you don’t want to preface the docker command with `sudo`, create a group called `docker` and add users to it.

To create the `docker` group and add your user:

1. Create the docker group.

```bash
sudo groupadd docker
```

Sometimes the `docker` group might already exist after the installation of Docker. This is okay, just move on to the next step.
2. Add your user to the docker group.

```
sudo usermod -aG docker $USER
```

Replace $USER with your user name.

3. Log out and log back in so that your group membership is re-evaluated. On Linux, you can also run the following command to activate the changes to groups:

```
newgrp docker
```

4. Verify that you can run docker commands without sudo.

```
docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints an informational message and exits.

More information on managing Docker as a non-root user can be found here: https://docs.docker.com/engine/install/linux-postinstall/

- Pull the Real-ESSI image

```
docker pull realessi/real-essi-repo:tag
```

Replace tag with the latest version of Real-ESSI. For example, the latest version is 21.01, then the pull command is docker pull realessi/real-essi-repo:21.01.

Current Real-ESSI Simulator version is kept up to date at the Real-ESSI web site HERE. In addition, you can find tags of Real-ESSI at https://hub.docker.com/repository/docker/realessi/real-essi-repo/tags.

- Run the Real-ESSI image:

```
docker run -it --rm -v your_working_directory:/workspace realessi/real-essi-repo:tag
```

Again, replace tag with the version of Real-ESSI you pulled. Once you start running the Real-ESSI Docker image, you are working inside the container. The container is Ubuntu 18.04 with Real-ESSI installed. Note that you should replace your_working_directory with the absolute path of your working directory.

- Run Real-ESSI:
The current directory on your local machine is shared with the container, so it can work with any files there. The files need to have the correct permissions to be run by a non-administrator user. You can move files after the container started and they will be recognized by the container.

After the simulation is finished, simply exit the container. You will see the output files and log file in your current directory. They will not be erased when you exit the container.

210.3.3 Performance of Real-ESSI Container

To test the performance of Real-ESSI container, a series of sequential and parallel simulations are conducted. The results and comparison are summarized in Figure 210.1.

210.4 Real-ESSI Simulator System Install

In addition to the Real-ESSI Program, Real-ESSI Simulator system consists of a pre-processing modules and post-processing modules. Installation of pre-processing modules is described in Chapter 207, on page 1176 in Jeremić et al. (1989-2021). Installation of post-processing modules is described in Chapter 208, on page 1243 in Jeremić et al. (1989-2021).

Both pre and post processing manuals are also available through the main Real-ESSI Simulator website: http://real-essi.info/.

210.4.1 Student Manual for Real-ESSI Simulator System Install

Students at ETH, Mr. Max Sieber and Mr. Antonio Felipe Salazar created a manual for installation of the Real-ESSI Simulator system on virtual machine computers. The manual is available HERE.
Figure 210.1: Comparison of Real-ESSI performance on local Linux machine and Linux/Windows containers.
Chapter 211

Software Platform Cloud Computing

(In collaboration with Dr. Yuan Feng, Dr. Han Yang, and Dr. Hexiang Wang)
211.1 Chapter Summary and Highlights

Described in this chapter are details of accessing and using Real-ESSI Simulator using remote computers, the so called "cloud" computational resources. Current focus is on using Amazon Web Services (AWS) computers.

211.2 Real-ESSI Cloud Computing Overview

Cloud computing refers to the accessing and computing over the Internet rather than on local computers. Cloud computing is a model for enabling on-demand access to a shared pool of configurable computing resources, which can be setup and released rapidly.¹

Using Real-ESSI Cloud Service, users can get computing instances on demand without requiring a lot of maintenance and financial resources a common, local parallel computer, cluster would require. In addition, users do not need to go through the installation of the dependent libraries, source-code compilation and the installation of other related software, for example preprocessing and post-processing environment. The complete Real-ESSI Simulator system is pre-configured and built within the image such that Real-ESSI Simulator system is portable over the cloud. A stable, release version of Real-ESSI is built and can be used anywhere and anytime.

There are two ways to obtain a Real-ESSI image on Amazon Web Services (AWS):

- Obtain a Real-ESSI private image from Prof. Boris Jeremić, see Section 211.3.1 on page 1341.
- Use a public image of Real-ESSI on AWS marketplace, as described in Section 211.3.2 on page 1353.

After a Real-ESSI image is launched, a Real-ESSI EC2 instance is generated on AWS. The instance can be accessed through a X2GO client. The procedures are written in Section 211.4 on page 1353.

When the simulation on the Real-ESSI instance is finished and all the output result files are fetched, remember to terminate the running instance so that AWS would not keep charging you. Section 211.5 on page 1356 describes how to terminate a running Real-ESSI instance. See Section 211.8 on page 1359 for more information about the cost of AWS cloud computing services.

211.2.1 Real-ESSI Cloud Service Content

One image is built for a single-machine setup, which contains

¹This is an excerpt from Jeremić et al. (1989-2021)
• Ubuntu 16.04 LTS Desktop and X2GO Server
• Real-ESSI sequential program
• Real-ESSI parallel program
• Real-ESSI 3C seismic motion developments (SW4)
• Real-ESSI pre-processing (gmESSI)
• Real-ESSI post-processing (PVESSIReader)
• Real-ESSI Editor, Sublime plug-ins
• Real-ESSI Documentation
• Real-ESSI Examples
211.3 Launch Real-ESSI Instance on AWS

A Real-ESSI instance can be launched either from the private image with authorization of Prof. Boris Jeremić or from the public image on AWS market place.

211.3.1 Launch Real-ESSI Instance from AWS Private Images

Follow the steps below to launch instances from Real-ESSI Private Image.

1. Create an AWS account.

AWS is the most widely used cloud service provider. If you do not have one, creating an AWS account is easy. You can create an AWS account through their website https://aws.amazon.com/. After you login, you can see the services on AWS Console Home as follows.

![AWS Console Home](image)

Figure 211.1: AWS Console Home.
2. Request the Real-ESSI image.

Real-ESSI image is currently a private Amazon Machine Images (AMI). After you get the 12-digit AWS account ID, email the AWS account ID to Prof. Boris Jeremić to obtain the Real-ESSI image.

From AWS Console Home, go to Services → EC2

![AWS Services](image_url)

Figure 211.2: AWS Services.
From EC2 Dashboard, go to AMIs to check the Real-ESSI image.

![Figure 211.3: AWS EC2 Dashboard AMIs.](image)

If users cannot find the Real-ESSI image, please make sure you are in the same AWS region with Prof. Boris Jeremić, the region is shown in the top-right corner on EC2 dashboard. The current Real-ESSI AMIs region are in both North California and Oregon.
3. Launch the Real-ESSI image.

![AWS EC2 Private AMIs](image)

Figure 211.4: AWS EC2 Private AMIs.
Follow the steps below to launch instances from the Real-ESSI image.

(a) Choose AMI.

Figure 211.5: EC2 Launch Steps: Choose AMI.
(b) Choose Instance Type

From AMIs, users can launch any number and type of instances and choose the desired EC2 configurations. In order to have the best experiences, the compute-optimized instances (C4, C5 as the latest one, as of early 2019) are recommended.

![EC2 Launch Steps: Choose Instance Type.](image)

Figure 211.6: EC2 Launch Steps: Choose Instance Type.
(c) Configure Instance

![Configure Instance Details](image)

**Figure 211.7:** EC2 Launch Steps: Configure Instance.
(d) Add Storage

Figure 211.8: EC2 Launch Steps: Add Storage.
(e) Add Tags

Figure 211.9: EC2 Launch Steps: Add Tags.
(f) Configure Security Group.
   Please keep the default Security Group setting.

(g) Review
   You may be asked to create a key-pair for later access of the instance you created. The key-pair can be reused later if you created other instances. Besides, the key-pair is portable across other machines. Last but not least, the key-pair cannot be recreated after you launch the instance, so please make sure you save the key-pair in a safe place.

![Step 7: Review Instance Launch]

Figure 211.10: EC2 Launch Steps: Review.
4. Check the launched instances

After the launch, you can view the running instance through EC2 Dashboard → Instances.

![EC2 Running Instances](image)

Figure 211.11: EC2 Running Instances.

You can login to your instances either by ssh or by using X2GO client 211.4. Please note that every time when you restart the instances, the public IP address will change.
5. Fix Public IP Address (Optional)

The public IP address of Real-ESSI instances change for each reboot. If users want to have a fixed public IP address for every login, users can allocate one elastic IP address and associate the IP address to a Real-ESSI instance such that users can have a fixed public IP address for each login.
6. Attach more Storage (Optional)

The Real-ESSI Image holds 30GB Hard disk and already uses 15GB. In the case of a real large simulations, this size hard drive might not be enough for the full output. Users can attach more storage through elastic block store.

211.3.2 Launch Real-ESSI Instance from AWS Market Place

This section gives a quick start guide for using Real-ESSI on AWS market place.

Real-ESSI Simulator system (pre processing, main Real-ESSI program, post processing) is available on Amazon Web Services MarketPlace. Point your web browser to the Amazon Web Services Market Place, and search for ”Real ESSI”, ”Real-ESSI” or ”MS ESSI”.

In summary, a quick guide to launching an instance from AWS Market Place is:

- Go to the ESSI Cloud Product Page.
- Click Continue to go to Launch ESSI from the Cloud.
- Click Manual Launch (use 1-Click Launch, if comfortable with settings).
- Click Launch from the EC2 Console for your preferred region.
- Select your preferred instance from the table, e.g. t2.micro.
- Click Review and Launch.

211.4 Connect to Real-ESSI Instance on AWS

211.4.1 Install X2GO Client

Before connecting to the Real-ESSI cloud, users should install the client-side of X2GO. X2Go is a remote desktop software that can visualize the launched Real-ESSI instance. Installation of X2GO for different operating systems is fairly straightforward, and users can find installation instructions on their own or follow installation instructions below.

211.4.1.1 Installing X2GO client on Ubuntu Linux

User can directly install X2GO client by using debian install utility, to install x2goclient.
211.4.1.2 Installing X2GO client on Apple Mac

Users can download the package through this link: http://code.x2go.org/releases/X2GoClient_latest_macosx_10_9.dmg.

211.4.1.3 Installing X2GO client on Windows

Users can download the package through this link: http://code.x2go.org/releases/X2GoClient_latest_mswin32-setup.exe.

211.4.1.4 Installing X2GO client on other operating systems

If you are using a different operating system, please refer to X2GO website for the installation. The X2GO website for client installation is https://wiki.x2go.org/doku.php/download:start
211.4.2 Configure the Client-Side of X2GO

For all operating systems, users will see the same session when they open the x2goclient new-session, as shown in Fig. 211.12.

![Configuration of X2GO client](image)

Figure 211.12: Configuration of X2GO client.

1. Users can name their own session.

2. AWS IP address is to be copied from EC2 management console, from the description TAB of launched instance, at the bottom of the page. This is IPv4 Public IP... it goes into Host: ...

3. AWS User Name is "ubuntu".

4. AWS ssh-key is the one saved from before, in .ssh directory

5. Please check the auto-login.

6. Please change the session type to XFCE.

7. Click OK to finish the configuration.

In addition to the Desktop login, users can also use ssh to login the Real-ESSI Terminal.

```
1. chmod 400 your_ssh_key.pem
2. ssh -i your_ssh_key.pem ubuntu@your.AWS.public.IP_address
```
211.4.3 Connect to the Launched Instance

Click the configured session to connect to the ESSI instance. You should see a virtual desktop pop up on your local machine, as shown in Fig. 211.13. Now you have successfully connected to the Real-ESSI Simulator instance on AWS. You can now use Real-ESSI Simulator within the virtual desktop.

![Connected to the Real-ESSI instance](image)

Figure 211.13: Connected to the already launched Real-ESSI instance.

211.5 Terminate Real-ESSI Instance on AWS

Once the Real-ESSI simulation on AWS is finished, the user can transfer output files to the local computer, or leave them on AWS, preferably on cheap S3 storage Section 211.8 on page 1359 provides detailed description of storage and transfer options and costs. **NOTE:** Users need to terminate the running Real-ESSI instance on AWS to avoid additional charges. The terminate operation is done on AWS console that is the same place where you launch the Real-ESSI instance. As shown Fig. 211.14, following steps are required:

1. Click ‘Instances’ from the sidebar to see all your running instances on AWS.

2. Choose the instance you want to terminate.

3. Click ‘Actions’.
4. Click ‘Instance State’

5. Click ‘Terminate’

![Terminate a Real-ESSI Simulator instance.](image)

Figure 211.14: Terminate a Real-ESSI Simulator instance.

211.6 Adding Permission for Private Real-ESSI Image to User AWS Accounts

login to AWS

sign in to console

go to image in a region, say N, California

then go to EC2

go to AMIs on left side

select image to be shared

go to Actions

go to Modify Image Permissions and put user account number then click Add Permission and then Save...
211.7 Real-ESSI Instructional Videos Cloud Computing

This section presents few short instructional videos about how to use Real ESSI on Amazon Web Services (AWS) computers.

211.7.0.1 Installing X2GO for Windows

Youtube instructional video.

211.7.0.2 Installing X2GO for Macintosh

Youtube instructional video.

211.7.0.3 Installing X2GO for Linux

Youtube instructional video.

211.7.0.4 Launch AWS Marketplace

Youtube instructional video.

211.7.0.5 Access Running Instance on AWS

Youtube instructional video.

211.7.0.6 Start Real-ESSI Program on AWS

Youtube instructional video.

211.7.0.7 Run Real-ESSI Example Model on AWS

Youtube instructional video.

211.7.0.8 Visualize Real-ESSI Example Model on AWS

Youtube instructional video.

211.7.0.9 Post-Process, Visualize Real-ESSI Results on AWS

Youtube instructional video.
211.8 Cost of AWS EC2

The cost breakdown for using Real-ESSI on AWS (EC2) is:

- **AWS computer cost**
  
  There are 3 ways to pay for AWS computer cost (EC2 instances):
  
  - **On-Demand instance**, offers a real, instant pay-per-use model. On-Demand instance is sold at a fixed price, and AWS computer availability is guaranteed (within the limits of the service-level agreement). **Running Real-ESSI On-Demand Instance**: User prepares simulation runs, and then can simulate problems at hand immediately.
  
  - **Spot instance**, uses spare AWS computers that users can bid for. Prices for those spot instances fluctuate based on the supply and demand of available AWS computers. When a user makes a bid for a Spot instance, a spot instance is launched when the bid exceeds the current Spot market price, and continues until terminated by the user. The user is charged the Spot market price, not the bid price while the instance runs. Spot instances can offer substantial savings over On-Demand instances, as shown in the AWS Spot Bid Advisor. **Running Real-ESSI using Spot instance**: User can prepare simulation runs, and then bid on computer hardware and run simulations at later time, when cost is acceptable.
  
  - **Reserved instance**, uses spare AWS computers during scheduled, later time as determined by AWS and reserved by the user. **Running Real-ESSI using Reserved Instance**: User prepares simulation runs, and then reserves AWS computer to simulate problem at hand at predetermined/reserved time.

- **AWS data storage cost**
  
  Input data/files and output data/files are stored using:
  
  - **Amazon Elastic Block Store (EBS)**, attached to a AWS computer (EC2 instance) during simulation run. Storage cost is charged by the size of storage in GB per month, pro-rated to the hour, until the storage is released. The cost of EBS is typically $0.10 per GB per month. When running Real-ESSI program on AWS computer, the storage is used during simulation, while the data (input and output) is transferred out of the AWS computer, to other type of storage that is less expensive (the so called S3 storage, see below), or to user’s desktop computer, before AWS computer/instance is terminated and storage released.
- **Amazon Simple Storage Service (S3)**, offers better value for longer term data storage. S3 pricing varies by region and frequency of access. Cost of S3 storage is typically between $0.0125 and $0.03 per GB per month.
- **Amazon Glacier**, provides storage at an even lower cost of $0.007 per GB per month for data archiving.

- **AWS data transfer cost**
  
  Data transfer charges are listed as part of the On-Demand EC2 pricing. Transfer is typically charged at $0.09 per GB beyond the first 1GB of data and up to the first 1TB of a given month. After the first TB, price drops down.

- **Real-ESSI program cost**
  
  Use of Real-ESSI for educational purposes is free. For commercial use of Real-ESSI, please contact Prof. Jeremić or one of the commercial companies that offer access to Real-ESSI on AWS.
211.8.1 Cost of Running Real-ESSI on AWS

211.8.1.1 Small Size Real-ESSI Example

Imposed Motion  Real-ESSI modeling and simulation on AWS summary:

- DOFs in the Model: 5,000
- Number of Time Step: 210
- Running Time: 30 Second
- Disk Space: 25 MB.
- Recommended Machine: Free Instance Amazon EC2 t2.micro

The Real-ESSI input files for this example are available HERE. The compressed package of input files is HERE.

The Modeling parameters are listed below:

- Elastic Material Properties
  - Mass density, $\rho$, $2000\ kg/m^3$
  - Shear wave velocity, $V_s$, $500\ m/s$
  - Young’s modulus, $E$, $1.1\ GPa$
  - Poisson’s ratio, $\nu$, $0.1$

The thickness of the shell structure is 2 meters. The simulation model is shown below.

The simulation results:
- The time series of simulation results is shown in Fig. 408.22.
- The response spectrum of motion is shown in Fig. 408.23.
Figure 211.15: Simulation Model.

Figure 211.16: Simulation Results.
Figure 211.17: Simulation Results: Acceleration Time Series with 1C imposed motion.
Figure 211.18: Simulation Results: Response Spectrum of Structure Top with 1C imposed motion.
Eigen Analysis  Real-ESSI modeling and simulation on AWS summary:

- DOFs in the Model: 5,000
- Number of Eigenmodes: 10
- Running Time: 3 Second
- Disk Space: 25 MB.
- Recommended Machine: Free Instance Amazon EC2 t2.micro

The Real-ESSI input files for this example are available HERE. The compressed package of input files is HERE.

The thickness of the shell structure is 2 meters. The simulation model is shown below.

![Simulation Model](image)

Figure 211.19: Simulation Model.

The eigen results:
Figure 211.20: Eigen Results (Eigen Mode 1 to 3 from left to right).

Figure 211.21: Eigen Results (Eigen Mode 4 to 6 from left to right).
211.8.1.2 Medium Size Real-ESSI Example

**Elastic Material** The compressed package of input files is available [HERE](#).

Real-ESSI modeling and simulation on AWS summary:

- DOFs in the Model: 132,000
- Number of Time Steps: 210
- Running Time: 10 minutes
- Disk Space: 3GB
- Recommended Machine: Amazon EC2 c4.2xlarge instance 8 cores.

- Estimated Bill in AWS Region Oregon/Ohio/Northern Virginia:
  - For simulation time: $0.398 \times 10 / 60 = $0.07
  - For General Purpose (SSD) Storage: $0.1 \times 3 = $0.3 (monthly)
  - For S3 Storage: $0.023 \times 3 = $0.069 (monthly)

The Modeling parameters are listed below

- **Elastic Material Properties**
  - Mass density, $\rho$, 2000 kg/m$^3$
  - Shear wave velocity, $V_s$, 500 m/s
  - Young’s modulus, $E$, 1.1 GPa
  - Poisson’s ratio, $\nu$, 0.1

The illustration results of the simulation is shown in Fig. 408.12. It is noted that outside the DRM layer, there are no outgoing waves.

**von-Mises Armstrong-Frederick Material** The compressed package of input files is available [HERE](#).

Real-ESSI modeling and simulation on AWS summary:

- DOFs in the Model: 132,000
- Number of Time Steps: 210
- Running Time: 46 minutes
Figure 211.22: Simulation Model.

Figure 211.23: Simulation Model.
• Disk Space: 3GB

• Recommended Machine: Amazon EC2 c4.2xlarge instance 8 cores.

• Estimated Bill in AWS Region Oregon/Ohio/Northern Virginia:
  
  - For simulation time: $0.398 \times 46/60 = $0.31
  
  - For General Purpose (SSD) Storage: $0.1 \times 3 = $0.3 (monthly)
  
  - For S3 Storage: $0.023 \times 3 = $0.069 (monthly)

The Modeling parameters are listed below

• von-Mises nonlinear hardening material model
  
  - Mass density, $\rho$, 2000 kg/m$^3$
  
  - Shear wave velocity, $V_s$, 500 m/s
  
  - Young’s modulus, $E$, 1.1 GPa
  
  - Poisson’s ratio, $\nu$, 0.1
  
  - von Mises radius, $k$, 60 kPa
  
  - Nonlinear kinematic hardening, $H_a$, 30 MPa
  
  - Nonlinear kinematic hardening, $C_r$, 60
  
  - Shear strength ($\approx \sqrt{2/3} H_a/C_r$), $S_u$, 408 kPa
  
  - Isotropic hardening rate, $K_{iso}$, 0 Pa

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 46 minutes.
211.8.1.3 Large Example

Elastic Simulation  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Real-ESSI modeling and simulation on AWS summary:

- DOFs in the Model: 210,000
- Number of Time Steps: 2065
- Running Time: 17 hours
- Disk Space: 45GB
- Recommended Machine: Amazon EC2 c4.8xlarge instance 36 cores.
- Estimated Bill in AWS Region Oregon/Ohio/Northern Virginia:
  - For simulation time: $1.591 \times 17 = $27.05
  - For General Purpose (SSD) Storage: $0.1 \times 45 = $4.5 \text{ (monthly)}
  - For S3 Storage: $0.023 \times 45 = $1.035 \text{ (monthly)}
  - For Network Bandwidth if transfer: $0.09 \times 45 = $4.05

SIMULATION TIME: With 32 cores on AWS EC2 c4.8xlarge instance, the running time for this example is 17 hours.

The Modeling parameters are listed below

- Soil
  - Unit weight, $\gamma$, 21.4 kPa
  - Shear velocity, $V_s$, 500 m/s
  - Young’s modulus, $E$, 1.3 GPa
  - Poisson’s ratio, $\nu$, 0.25
  - Shear strength, $S_u$, 650 kPa
  - von Mises radius, $k$, 60 kPa
  - kinematic hardening, $H_0$, 30 MPa
  - kinematic hardening, $C_r$, 25

- Structure
Figure 211.24: Simulation Model.

- Unit weight, \( \gamma \), 24 kPa
- Young’s modulus, \( E \), 20 GPa
- Poisson’s ratio, \( \nu \), 0.21

The input motion is a 3C wave from SW4.
Inelastic Simulation  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Real-ESSI modeling and simulation on AWS summary:

- DOFs in the Model: 210,000
- Number of Time Steps: 2065
- Running Time: 30 hours
- Disk Space: 45GB
- Recommended Machine: Amazon EC2 c4.8xlarge instance 36 cores.
- Estimated Bill in AWS Region Oregon/Ohio/Northern Virginia:
  - For simulation time: $1.591 \times 30 = $47.73
  - For General Purpose (SSD) Storage: $0.1 \times 45 = $4.5 (monthly)
  - For S3 Storage: $0.023 \times 45 = $1.035 (monthly)
  - For Network Bandwidth if transfer: $0.09 \times 45 = $4.05

SIMULATION TIME: With 32 cores on AWS EC2 c4.8xlarge instance, the running time for this example is 30 hours.

The Modeling parameters are listed below

- Soil
  - Unit weight, $\gamma$, 21.4 kPa
  - Shear velocity, $V_s$, 500 m/s
  - Young’s modulus, $E$, 1.3 GPa
  - Poisson’s ratio, $\nu$, 0.25
  - Shear strength, $S_u$, 650 kPa
  - von Mises radius, $k$, 60 kPa
  - kinematic hardening, $H_a$, 30 MPa
  - kinematic hardening, $C_r$, 25

- Structure
- Unit weight, $\gamma$, 24 kPa
- Young’s modulus, $E$, 20 GPa
- Poisson’s ratio, $\nu$, 0.21

- **Contact**
  - Initial axial stiffness, $k_n^{init}$, 1e9 N/m
  - Stiffening rate, $S_r$, 1000 /m
  - Maximum axial stiffness, $k_n^{max}$, 1e12 N/m
  - Shear stiffness, $k_t$, 1e7 N/m
  - Axial viscous damping, $C_n$, 100 $N \cdot s/m$
  - Shear viscous damping, $C_t$, 100 $N \cdot s/m$
  - Friction ratio, $\mu$, 0.25

Figure 211.25: Simulation Model.
211.8.2 AWS for Education

Amazon Web Services provides grants for educators and students from member institution\(^2\) through AWS Educate program. AWS Educate offers cloud content, training, collaboration tools and AWS technology at no cost. Some of the AWS Educate program benefits:

- **For Educators**
  - $200 in AWS credits per educator - at member institutions.
  - $75 in AWS credits per educator - at non-member institutions.
  - Free AWS Technical Essentials eLearning course.
  - Free access to AWS content for classes.

- **For Students**
  - $100 in AWS credits per student - at member institutions.
  - $40 in AWS credits per student - at non-member institutions.
  - Access to AWS Technical Essentials Training Course (a $600 value).

If you have an email address from an educational institutions, you can use Real-ESSI on AWS for free through AWS Educate.

\(^2\)List of member institution is available at this [LINK](#).

211.8.3 AWS for Government

211.8.3.1 AWS GovCloud

211.8.3.2 AWS Secret Region
Chapter 212

Hardware Platform Design and Development

212.1 Chapter Summary and Highlights

212.2 Introduction

Parallel computer used for simulations is based on a Beowulf concept (Sterling et al., 1995; Reschke et al., 1996; Sterling et al., 1998, 1999; Warren et al., 1998; Ridge et al., 1997). Hardware for a specific application to parallel computing for elastic-plastic finite elements has gone through a number of iterations (Jeremić et al., 1998, 1999) and is still evolving as the hardware components change in time and as our algorithms change/improve.

The choice of hardware platform is for a cost effective, off the shelf PC components, with about 2GB of memory per CPU/core and plenty of disk space (about 4TB for large dynamic runs). Linux is chosen as an operating system as it offers the best performance, is available in open source, which ensures that significant number of developers can contribute their expertise and can be customized to suite the needs of a parallel hardware and software. Microsoft Windows or Apple IOS operating systems are not best suited for parallel computing as their main development goal is user friendliness and not efficiency.

212.3 The NRC ESSI Computer

212.3.1 Version: December 2010

Operating System: Linux Fedora Core 14.

Kernel: 2.6.35.10-74.fc14.x86_64

Compute Nodes (two):

- CPU: 2 × Intel Xeon E5620 Westmere 2.4 GHz Quad Core (8 threads) 32nm CPU with 256 KB Cache/core and 12MB Shared L3, DDR3-1066, 5.86GT/sec QPI, 80W
- RAM: 6 × 4GB DDR3 1333 MHz ECC/Registered Memory (24GB Total Memory 1066MHz)
- Disk: 8 × 500 GB Seagate Constellation ES 3.5" SATA/300 ST3500514NS 32MB Cache, 3Gb/s, NCQ, 7200RPM, 1.2 million hours MTBF Maximum Sustained Transfer Rate: 140 MB/sec (Linux Software RAID10)

Network: single GigaBit

212.3.2 Version: April 2012

In addition to the previous version.
Operating System:
Kernel:
Controller Node 1 (one):
- CPU: 2 x Opteron 6234 (2.4GHz, 12-Core, G34, 16MB L3 Cache) 115W TDP, 32nm
- RAM: 32GB (8 x 4GB) Operating at 1333MHz Max (DDR3-1333 ECC Registered DIMMs)
- NICS: Integrated Intel 82576 Dual-Port Gigabit Ethernet Controller
- Disk: 8 x 2TB Toshiba MK2002TSKB (3Gb/s, 7.2K RPM, 64MB Cache) 3.5” SATA

Compute Nodes, 8 (eight):
- CPU: 2 x Opteron 6234 (2.4GHz, 12-Core, G34, 16MB L3 Cache) 115W TDP, 32nm
- RAM: 32GB (8 x 4GB) Operating at 1333MHz Max (DDR3-1333 ECC Registered DIMMs)
- NICS:
  - Intel 82576 Dual-Port Gigabit Ethernet Controller
  - InfiniBand: ConnectX-2 QDR IB 40Gb/s Controller with QSFP Connector
- Disk: 1TB Toshiba MK1002TSKB (3Gb/s, 7.2K RPM, 64MB Cache) 3.5” SATA

Network (dual):
- HP ProCurve Switch 1810-48G 48 Port 10/100/1000 ports Web Managed Switch
- IB Switch: Mellanox MIS5030Q-1SFCA 36-port QDR switch; Cables: 9 x 3mtr QSFP-QSFP - Rating: QDR
Part 300

Verification and Validation
Chapter 301

Verification and Validation Introduction

301.1 Chapter Summary and Highlights

301.2 Verification and Validation

Suggested reading:

Roache (1998); Oberkampf et al. (2002); Oberkampf (2003); Oberkampf and Trucano (2008); Oberkampf and Roy (2010); Roy and Oberkampf (2011); Oden et al. (2005); Babuška and Oden (2004); Babuska et al. (2004); Oden et al. (2010a); Oden et al. (2010b);

- How do we use experimental simulations to develop and improve models
- How much can (should) we trust model implementations (verification)
- How much can (should) we trust numerical simulations (validation)

301.2.1 Trusting Simulation Tools

- **Verification**: The process of determining that a model implementation accurately represents the developer’s conceptual description and specification. Mathematics issue. Verification provides evidence that the model is solved correctly.

- **Validation**: The process of determining the degree to which a model is accurate representation of the real world from the perspective of the intended uses of the model. Physics issue. Validation provides evidence that the correct model is solved.

301.2.2 Importance of V & V

- V & V procedures are the primary means of assessing accuracy in modeling and computational simulations
- V & V procedures are the tools with which we build confidence and credibility in modeling and computational simulations

301.2.3 Maturity of Computational Simulations

NRC committee (1986) identified stages of maturity in CFD

- **Stage 1**: Developing enabling technologies (scientific papers published)
- **Stage 2**: Demonstration of and Confidence in technologies and tools (capabilities and limitations of technology understood)
- **Stage 3:** Compilation of technologies and tools (capabilities and limitations of technology understood)

- **Stage 4:** Spreading of the effective use (changes the engineering process, value exceeds expectations)

- **Stage 5:** Mature capabilities (fully dependable, cost effective design applications)
301.2.4 Role of Verification and Validation

![Diagram of the role of verification and validation](image1)

**Figure 301.1**: Role of Verification and Validation per Oberkampf et al. (2002).

![Diagram of the role of verification and validation](image2)

**Figure 301.2**: Role of Verification and Validation per Oden et al. (2010a).
301.2.4.1 Alternative V & V Definitions

IEEE V & V definitions (1984):

- **Verification**: The process of determining whether the products of a given phase of the software development cycle fulfill the requirements established during the previous phase.

- **Validation**: The process of evaluating software at the end of the software development process to ensure compliance with software requirements.

- Other organizations have similar definitions:
  - Software quality assurance community
  - American Nuclear Society (safety analysis of commercial nuclear reactors)
  - International Organization for Standardization (ISO)

301.2.4.2 Certification and Accreditation

- **Certification**: A written guarantee that a system or component complies with its specified requirements and is acceptable for operational use (IEEE (1990)).
  - Written guarantee can be issued by anyone (code developer, code user, independent code evaluator)
  - Code certification is more formal than verification and validation documentation

- **Accreditation**: The official certification that a model or simulation is acceptable for use for a specific purpose (DOD/DMSO (1994))
  - Only officially designated entities can accredit
  - Normally appointed by the customers/users of the code or legal authority
  - Appropriate for major liability or public safety applications

301.2.4.3 Independence of Computational Confidence Assessment

1. V&V conducted by the computational tool developer; *No Independence*

2. V&V conducted by a user from same organization

3. V&V conducted by a computational tool evaluator contracted by developer’s organization

4. V&V conducted by a computational tool evaluator contracted by the customer
5. V&V conducted by a computational tool evaluator contracted by the a legal authority; High Independence
301.2.5 Detailed Look at Verification and Validation

![Diagram of Verification and Validation process]

301.2.5.1 On Verification

**Verification**: The process of determining that a model implementation accurately represents the developer’s conceptual description and specification.

- Identify and remove errors in computer coding
  - Numerical algorithm verification
  - Software quality assurance practice
- Quantification of the numerical errors in computed solution
301.2.5.2 On Validation

Validation: The process of determining the degree to which a model is accurate representation of the real world from the perspective of the intended uses of the model.

- Tactical goal: Identification and minimization of uncertainties and errors in the computational model
- Strategic goal: Increase confidence in the quantitative predictive capability of the computational model

Goals of Validation  Quantification of uncertainties and errors in the computational model and the experimental measurements

- Goals on validation
  - Tactical goal: Identification and minimization of uncertainties and errors in the computational model
  - Strategic goal: Increase confidence in the quantitative predictive capability of the computational model

- Strategy is to reduce as much as possible the following:
  - Computational model uncertainties and errors
  - Random (precision) errors and bias (systematic) errors in the experiments
  - Incomplete physical characterization of the experiment
Validation Procedure Uncertainty

- Aleatory uncertainty → inherent variation associated with the physical system of the environment (variation in external excitation, material properties...). Also known as irreducible uncertainty, variability and stochastic uncertainty.

- Epistemic uncertainty → potential deficiency in any phase of the modeling process that is due to lack of knowledge (poor understanding of mechanics...). Also known as reducible uncertainty, model form uncertainty and subjective uncertainty.

Types of Physical Experiments

- Traditional Experiments
  - Improve the fundamental understanding of physics involved
  - Improve the mathematical models for physical phenomena
  - Assess component performance

- Validation Experiments
  - Model validation experiments
  - Designed and executed to quantitatively estimate mathematical model’s ability to simulate well defined physical behavior
  - The simulation tool (SimTool) (conceptual model, computational model, computational solution) is the customer

Validation Experiments

- A validation experiment should be jointly designed and executed by experimentalist and computationalist
  - Need for close working relationship from inception to documentation
  - Elimination of typical competition between each
  - Complete honesty concerning strengths and weaknesses of both experimental and computational simulations

- A validation Experiment should be designed to capture the relevant physics
  - Measure all important modeling data in the experiment
Characteristics and imperfections of the experimental facility should be included in the model.

- A validation experiment should use any possible synergism between experiment and computational approaches.
  - Offset strength and weaknesses of computations and experiments.
  - Use high confidence simulations for simple physics to calibrate or improve the characterization of the experimental facility.
  - Conduct experiments with a hierarchy of physics complexity to determine where the computational simulation breaks (remember, SimTool is the customer!)

- Maintain independence between computational and experimental results.
  - Blind comparison, the computational simulations should be predictions.
  - Neither side is allowed to use fudge factors, parameters.

- Validate experiments on unit level problems, hierarchy of experimental measurements should be made which present an increasing range of computational difficulty.
  - Use of qualitative data (e.g. visualization) and quantitative data.
  - Computational data should be processed to match the experimental measurement techniques.

- Experimental uncertainty analysis should be developed and employed.
  - Distinguish and quantify random and correlated bias errors.
  - Use symmetry arguments and statistical methods to identify correlated bias errors.
  - Make uncertainty estimates on input quantities needed by the SimTool.
301.3 Prediction

- Prediction: use of computational model to foretell the state of a physical system under consideration under conditions for which the computational model has not been validated.

- Validation does not directly make a claim about the accuracy of a prediction.
  - Computational models are easily misused (unintentionally or intentionally).
  - How closely related are the conditions of the prediction and specific cases in validation database.
  - How well is physics of the problem understood.

301.3.1 Relation Between Validation and Prediction

Quantification of confidence in a prediction:

- How do I quantify validation and its inference value in a prediction?
- How do I quantify verification and its inference value in a prediction?
- How far are individual experiments in my validation database from my physical system of interest?
301.4 Application Domain

- Rarely applicable to engineering systems (certainly not for infrastructure objects like bridges, buildings, port facilities, dams...)
- Even if the engineering system is small, environmental influences (generalized loads, conditions, wear and tare) are hard to predict
- Human factors (take Mars rover example with a memory overflow, operator forgot to flush the memory...)
- Inference ⇒ Based on physics or statistics
- Validation domain is actually an aggregation of tests and thus might not be convex (bifurcation of behavior)
- Experimental facilities provide validation domain that is (for the most part) exclusively non-overlapping with the application domain.

301.4.1 Importance of Models and Numerical Simulations

- Verified and Validated models can be used for assessing behavior of
  - components or
  - complete systems,
- with the understanding that the environmental influences cannot all be taken into the account prior to operation
- but with a good model, their influence on system behavior can be assessed as need be (before or after the event)
301.4.2 Prediction under Uncertainty

- Ever present uncertainty needs to be estimated for predictions
- Identify all relevant sources of uncertainty
- Create mathematical representation of individual sources
- Propagate representation of sources through modeling and simulation process (Probabilistic Elastic Plastic Theory)
Chapter 302

Source Code Verification

(In collaboration with Dr. Yuan Feng, and Dr. Han Yang)
302.1 Chapter Summary and Highlights

Hatton (1997); Roache (1998); Oberkampf et al. (2002); Oberkampf (2003); Oden et al. (2005); Babuška and Oden (2004); Oden et al. (2010a); Oden et al. (2010b);

302.1.1 Numerical Algorithm Verification

302.1.2 Software Quality Assurance
Chapter 303

Code Stability Verification

(In collaboration with Dr. Yuan Feng and Dr. Han Yang)
303.1 Chapter Summary and Highlights

303.2 Introduction to Code Stability

This activity addresses source code stability. Source code verification is addressed elsewhere.

303.3 Motivation

In the software development process, Real-ESSI program is a framework and new features are continuously added. From time to time, some revisions for one feature may affect the normal operation of other existing components, like a specific element or material. To guarantee the stability and correctness of Real-ESSI program for each update. A group of test cases are collected in an automatic test suite. Then, after each revision, the developers are required to successfully pass all the test cases before the revisions are finally accepted to the trunk branch.

The features of the automatic Real-ESSI test is listed below:

- Automatic comparison of the maximum displacement output between the original and the new ESSI results.
- Automatic comparison of all the displacement and stress/strain output between the original and the new ESSI results.
- Automatic comparison of the terminal output/log between the original and the new ESSI results.
- Relative difference between the original and the new ESSI results.
- Colorful diagnostic information in the terminal.
- Support for the report in HTML format.
- Version information of both the original and the new ESSI.
- Number of passed cases and the statistics.

303.4 The framework of the automatic test

In practice, all the test cases are collected in one main folder and each test case has an independent subdirectory. In addition, Bash and Python are employed go over each leaf directory of the test cases.
folder to execute essi and compare the results. A verification report is generated automatically after all
the test cases are executed.

Regarding the test cases for the version stability, it is not necessary to choose the great model with
lots of elements. The goal of version stability is to guarantee that the revision for one feature should not
affect the normal operation of other commands. So the selection rule of test cases is to cover as much
Real-ESSI DSL (domain specific language) commands as possible.

303.5 Installation and Tutorial

303.5.1 Installation

Makefile, Bash and Python are used to run all the test cases. So the automatic test is portable over
various Linux platform as long as Makefile, Bash and Python are available. Besides, git is also required
to download the test suite. In addition, if the user wants an additional report in HTML format, another
package called aha is required. Install aha package on Ubuntu by using this command.

```
1 ${Real-ESSI}/CompGeoMechUCD_Miscellaneous/examples/
```

Notes: the automatic test will call the executable essi in the system/user PATH. So please make
sure you have compiled or installed Real-ESSI first and then run this automatic test suite.

303.5.2 Tutorial

303.5.2.1 Run all verification test cases

In order to run all test cases to verify the installation, users can run

```
1 cd $RealESSI_PATH/
2 bash run_all_verification_sequential.sh $EXECUTABLE_PATH
3 bash run_all_verification_parallel.sh $EXECUTABLE_PATH
```

In addition to the conventional "-DDEBUG_MODE=DEBUG or OPTIMIZED", Users are required to
do the test for executables compiled using the compiler options "-DDEBUG_MODE=O1 or O2" to fully
verify the code. The verification will list all the results and errors. The errors may be big when the mesh
is too coarse, or when Poisson’s ratio is too high.

Furthermore, if developers want to verify against a previous version, developers can run

```
1 cd $RealESSI_PATH/
```
bash run_code_stability.sh $GitTAG

to test the verification results against one previous git version. The script will automatically checkout
to the previous git-tag in detached mode and compile the old essi. After running test cases, developers
can checkout to the original testing branch by running

cd $RealESSI_PATH/
git checkout $TestingBranch

In addition, if users want to clean the test results, users can run

cd $RealESSI_PATH/
bash clean_all_verification.sh

303.5.2.2 Run a single type of verification test cases

The usage of automatic test is written after the build process of source-code in Section 209.8.

For a single type of verification, for instance, in the folder

cd $RealESSI_PATH/CompGeoMechUCD_Miscellaneous/examples/analytic_solution

there are two clean options available in the main folder.

- The first one is make clean. This will only clean the new essi output results, including HDF5 files,
terminal logs, and comparison logs.

- The second one is make cleanall. This will clean both the new and old essi output results for
version stability test.

303.6 The underlying implementation of the automatic test

In most cases, the developers are not required to read and modify the underlying implementation of
the automatic test. However, a basic introduction to the underlying implementation will help the future
developers to customize the automatic test suite when necessary.

303.6.1 Generate the original results

When the command below is called,

bash generate_original.sh
Three things will be done. First, `make cleanall` will be called. This means that all the new and original output will be removed. Second, `essi` will be called to run each test cases and to generate the original HDF5 and original terminal output logs. Third, a bash command is employed to rename the HDF5 file and terminal log by adding `original` at the end of the filename.

### 303.6.2 Run essi and make comparison

When the command below is called,

```bash
bash make_comparison.sh
```

Four things will be done.

- First, `essi` will be called again to re-run each test cases. Please note that this `essi` should be the newly compiled `essi` in the development and debug stages.

- Second, a python-based comparison function will be called to compare the maximum displacement in the HDF5 output for each test case.

- Third, another python-based comparison function will be called to compare all the displacement and stress/strain results in the HDF5 output for each test case.

- Fourth, the terminal output log will be compared.

#### 303.6.2.1 The terminal output/log comparison

The motivation to compare the terminal output is to avoid the unnecessary debug-purpose messages in the terminal output. During the debug stages, developers usually print out the variable values in the terminal. However, in the production stages, the debug-purpose messages should be disabled.

During the terminal output/log comparison, some lines are always different for each `essi` execution. These lines should be removed from comparison to avoid the false mismatch.

- The first type of different lines are the version information. The `essi` version information includes the `essi` compile time and execution time. They are always different for each execution. Therefore, these lines are extracted and the version information is printed out at the end of the comparison report.

- The second type of different lines are the `ETA` information. `ETA` stands for estimated time of arrival, which is a prediction for the `essi` execution time. However, the `ETA` information is inaccurate and they are different for each execution. Therefore, the `ETA` information is not compared either.
303.6.2.2 Reduce the comparison items and comparison time

During the debug stages, the developers might only want to compare the HDF5 output and the developers want to keep the verbose terminal messages so developers do not want to compare the terminal output log. Besides, during the debug stages, the developers may not want the time-consuming HDF5 output comparison. Therefore, to reduce the comparison items and comparison time, the developers can comment out the last two lines in `make_comparison.sh`.

303.7 Report Sample

For the sake of convenience, automatic test provides colorful diagnostic information. The green color is for the passed (matched) test case, while the red color is for the failed (mismatched) test case. The illustrative results are shown below. In addition, the automatic test also reports the relative path (location) of the test case. So if one of the test cases failed, developers can locate the subdirectory easily and check the mismatched model.

303.7.1 Passed test case

![Passed test case](image1.png)

Figure 303.1: The report sample for a passed test case

303.7.2 Failed test case

In Figure 303.2, the location means the value location in the displacement results matrix of a HDF5 file. In the displacement results matrix, the column number is the step number and the row number is the dof (degree of freedom) number.
303.8 Future contribution

The automatic test is a test framework. It is easy to contribute your new test cases to the framework. The newly added test case must meet the following two requirements.

- The test case should be added as an independent leaf subdirectory within the `test_cases` folder.
- The test case should have a `main.fei` as the main model file.
Chapter 304

Verification and Validation for Constitutive Problems

(In collaboration with Dr. Yuan Feng and Dr. Han Yang)
304.1 Chapter Summary and Highlights

304.2 Verification of Constitutive Integration

In this section, the accuracy analysis of the implicit algorithm is assessed. Examples of simple models (von Mises and Drucker-Prager) for accuracy analysis are demonstrated to verify general implicit algorithm. Convergence performance analysis is conducted. More details on accuracy analysis and consistent tangent stiffness are explained. Numerical simulation examples are demonstrated using the implemented framework. Special concerns are on the comparison of experimental data and numerical results of Dafalias-Manzari model.

304.2.1 Error Assessment

There are various error measures for the integration algorithms. Simo and Hughes (1998), Manzari and Prachathananukit (2001) used the relative stress norm by Equation 304.1,

$$\delta^r = \frac{\sqrt{(\sigma_{ij} - \sigma^*_{ij})(\sigma_{ij} - \sigma^*_{ij})}}{\sqrt{\sigma^*_{pq}\sigma^*_{pq}}}$$  (304.1)

where $\sigma^*_{ij}$ is the ‘exact’ stress solution, and $\sigma_{ij}$ the calculated stress solution. Alternatively, Jeremić and Sture (1997) used the normalized energy norm by Equation 304.2,

$$\delta^n = \frac{\|\sigma_{ij} - \sigma^*_{ij}\|}{\|p^{unit}\|}$$  (304.2)

where $\|\sigma_{ij}\|^2 = \sigma_{ij}D_{ijkl}\sigma_{kl}$, and $D_{ijkl}$ is the elastic compliance fourth-order tensor, $p^{unit}$ is the ‘unit’ energy norm for normalization.

The relative stress norm by Equation 304.1 is more reasonable since two points having the same $\|\sigma_{ij} - \sigma^*_{ij}\|$ but different $\sigma^*_{pq}\sigma^*_{pq}$ should have different error measures. However, this norm becomes singular and possible meaningless when $\sigma^*_{pq}\sigma^*_{pq}$ close to zero. The normalized energy norm by Equation 304.2 have no such singularity problem but it may give the same error index for two points having the same $\|\sigma_{ij} - \sigma^*_{ij}\|$ but different $\sigma^*_{pq}\sigma^*_{pq}$. In this work, we use these two error measure methods, but for simplicity, Equation 304.2 is modified into

$$\delta^r = \frac{\sqrt{(\sigma_{ij} - \sigma^*_{ij})(\sigma_{ij} - \sigma^*_{ij})}}{\sqrt{\sigma^0_{pq}\sigma^0_{pq}}}$$  (304.3)

where $\sigma^0_{pq}\sigma^0_{pq}$ is evaluated at some non-zero initial isotropic stress state. That is, the normalized error is evaluated by Equation 304.3, and the relative error is evaluated by Equation 304.1.
In our examples, the initial stress state point is set \( p^0 = 100 \) kPa, \( q^0 = 0 \) kPa, \( \theta^0 = 0 \), which is the \( \sigma_{pq}^0 \) in Equation 304.3. The one-step predicted stress state point for the implicit algorithm is within the range of \( 0.1 \leq p \leq 100 \) kPa, \( 0 \leq q \leq 100 \) kPa, \( 0 \leq \theta \leq \pi/3 \). The ‘exact’ solution is actually unknown for most elastoplastic problems. Here the ‘exact’ solution is simply replaced by 50 substep solution of the explicit algorithm in the same one-step prediction incremental. All these error evaluations are within the material constitutive level.

The first test examples are von Mises models with the uniaxial yield strength \( k = 50 \) kPa, with linear elasticity parameters are Young’s modulus \( E = 1 \times 10^5 \) kPa, and Poisson’s ratio \( \nu = 0.25 \).

Figures 304.1 and 304.2 show the iso-error maps for the von Mises model with linear isotropic hardening. The linear hardening modulus \( H = 2 \times 10^4 \) kPa. The blue lines represents the yield surface boundary. It can be seen that the error magnitudes are as small as \( 10^{-10} \) to \( 10^{-9} \), which implies that the solutions by implicit algorithm for this linear isotropic hardening von Mises model are numerically accurate.

Figure 304.1: Normalized iso-error maps of von Mises model with linear isotropic hardening.

Figures 304.3 and 304.4 show the iso-error maps for the von Mises model with Armstrong-Frederick translational kinematic hardening. The hardening parameters are \( h_a = 5 \times 10^4 \) kPa and \( C_r = 2.5 \times 10^3 \). It can be seen that errors are very small which proves the good performance of the implicit algorithm. The iso-error map gives a good trend, i.e., the further away from the yield surface, the errors become more pronounced; the normalized errors are pressure-independent, which fits well the feature of von Mises model; the iso-error lines in the \( q - \theta \) figure are parallel to the yield surface and are independent of the Lode’s angle \( \theta \), which again fits well with von Mises model which is only \( q \)-related.

The second test examples are Drucker-Prager model with yield surface constant \( q/p = 0.8 \). Linear
elasticity parameters are Young's modulus $E = 1 \times 10^5$ kPa, and Poisson's ratio $\nu = 0.25$.

The iso-error maps for perfectly plastic Drucker-Prager model are shown in Figures 304.5 and 304.6. The blue lines represents the yield surface boundary. It can be seen that the error magnitudes are as small as $10^{-11}$ to $10^{-9}$. Again, these errors are so small that we can consider that the implicit algorithm give accurate solutions numerically.

Another Drucker-Prager model is with Armstrong-Frederick rotational kinematic hardening, and the parameters are $h_a = 20$, $C_r = 2$. The iso-error maps are shown in Figures 304.7 and 304.8. Unlike
von Mises model, the normalized errors are pressure-dependent, which fits well the feature of Drucker-Prager model; the iso-error lines in the $q - \theta$ figure are parallel to the yield surface and are independent of the Lode’s angle $\theta$, which still fits well with Drucker-Prager model which does not consider the third stress invariant, Lode’s angle $\theta$. From the relative iso-error maps in Figure 304.8, very dense iso-error lines are investigated in the region of small pressure, which is evidently due to the cone apex singularity of Drucker-Prager yield surface.

From the error analysis by the above von Mises and Drucker-Prager models, One finds that the implemented implicit algorithm can offer accurate solutions for simple models with simple hardening laws,

Figure 304.4: Relative iso-error maps of von Mises model with Armstrong-Frederick kinematic hardening.

Figure 304.5: Normalized iso-error maps of Drucker-Prager perfectly plastic model.
e.g. von Mises model with linear hardening and Drucker-Prager model with perfectly plastic hardening (no hardening). Complicated hardening laws increases the error even for simple plastic models, although the errors are still small. These observations match the well known conclusion that the error of the implicit algorithm is pretty dependent on the smoothness of the solution. The implemented implicit algorithm proves very robust for von Mises and Drucker-Prager model with simple or complicated hardening laws.

Figures 304.9 and 304.10 present the iso-error maps of Dafalias-Manzari model. The initial void ratio is 0.8, and the other parameters are from Dafalias and Manzari (2004a). The blue lines represents...
Figure 304.8: Relative iso-error maps of Drucker-Prager model with Armstrong-Frederick kinematic hardening.

the yield surface boundary (slope ratio \( m = 0.01 \)). Unlike von Mises and Drucker-Prager models, the iso-error lines in the \( q - \theta \) figure of Dafalias-Manzari model are not parallel to the yield surface and are dependent of the Lode’s angle \( \theta \), which was one of the highlighting improvements upon the previous version (Manzari and Dafalias, 1997). From Figure 304.10, when the predicted stress \( q \) close to 100 kPa, or or about 100 times the yield strain increment, the relative errors can reach up to 100%, which implies that even for implicit algorithm, Dafalias-Manzari model still requires small strain increments. However, when \( q < 30 \) kPa, or about 30 times of the yield strain increment, the relative errors are less than 5%, excepts at the region close to the yield surface apex.

It should be pointed out that errors for the complex Dafalias-Manzari model are much bigger than those of simple models (e.g. von Mises and Drucker-Prager), due to its high non-linearity. However, if the predicted stress (or in other words, the strain increment) is small enough, the algorithm errors are within a small tolerant range.

Figures 304.9 and 304.10 are based on an approach of averaged elastic moduli. Instead, Figures 304.11 and 304.12 present iso-error maps based on constant elastic moduli approach.

304.2.2 Constitutive Level Convergence

In the implemented implicit algorithm, the iteration continues until the absolute value of yield function and the residue norm of considering variables are less than some small tolerances, or if by equations,

\[
|f| \leq Tol1; \quad r_{norm} = \|r\| \leq Tol2
\]

(304.4)
Three examples including simple von Mises model with linear isotropic hardening, relative complicated Drucker-Prager model with Armstrong-Frederick kinematic hardening, and even more complicated Dafalias-Manzari model considering fabric dilation effect are presented here to show the constitutive level convergence performances for the implemented implicit algorithm. In all these examples, both $|f|$ and $r_{\text{norm}}$ v.s. iteration numbers are plotted. Iteration number 0 represents the ‘virtual’ iteration number before return mapping implicit iteration cycle. $|f|$ at iteration number 0 thus means $|f|$ at the first predicted stress for each load increment; there is no value of $r_{\text{norm}}$ at iteration number 0. A tolerance of $Tol_1 = Tol_2 = 1 \times 10^{-7}$ is for both $|f|$ and $r_{\text{norm}}$. The iteration stops when $|f| \leq Tol_1$.
and \( r_{\text{norm}} \leq T_{\text{ol}} \) are satisfied, even if there is only one iteration number. The initial stress is an isotropic stress state of \( p_0 = 100 \) kPa. The undrained-like load increment is adopted by strain control as \( \epsilon_{11} = -2\epsilon_{22} = -2\epsilon_{33} = n \times \Delta \epsilon \), where \( n \) is the load increment number and \( \Delta \epsilon \) is the strain increment interval, \( \epsilon_{ij} \) are strain components.

Figure 304.13 shows the typical constitutive level convergence performance for von Mises model with linear isotropic hardening. The input parameters are Young’s Modulus \( E = 1 \times 10^5 \) kPa, Poisson’s ratio \( \nu = 0.25 \), the material strength \( k = 50 \) kPa, and the linear isotropic hardening modulus \( H = 2 \times 10^4 \) kPa. The strain increment interval \( \Delta \epsilon \) is set \( 2 \times 10^{-4} \). It can be seen that for this simple example, only
two iteration steps are needed and $|f|$ and $r_{\text{norm}}$ are far smaller than the tolerances and in fact close to the machine floating error value, or in other words, the stresses are exactly at the yield surface and the residue norm is zero.

![Graph showing typical convergence for von Mises model with linear isotropic hardening](image)

Figure 304.13: Typical convergence for von Mises model with linear isotropic hardening (tolerance value $1 \times 10^{-7}$).

Figure 304.14 shows the typical constitutive level convergence performance for Drucker-Prager model with Armstrong-Frederick kinematic hardening. The input parameters are Young’s Modulus $E = 1 \times 10^4$ kPa, Poisson’s ratio $\nu = 0.25$, the material $q/p$ ratio is 0.8, and the Armstrong-Frederick parameters are $h_a = 20$, $C_r = 2$. The strain increment interval $\Delta \epsilon$ is set $-2 \times 10^{-4}$. For this example, both $|f|$ and $r_{\text{norm}}$ are stably decreasing with the increasing iteration number; However, $|f|$ and $r_{\text{norm}}$ show different rates; $|f|$ needs 5 iteration steps while $r_{\text{norm}}$ needs 7 iteration steps; The convergence rate of $r_{\text{norm}}$ lags behind that of $|f|$.

Figure 304.15 shows the typical constitutive level convergence performance for the complicated Dafalias-Manzari model considering fabric dilation effect. The input parameters are as in Table 304.1, and the initial void ration is set as 0.8. Different from the above examples, The strain increment interval $\Delta \epsilon$ is set a much smaller value of $-1 \times 10^{-5}$. In this example, again, both $|f|$ and $r_{\text{norm}}$ are stably decreasing with the increasing iteration number; However, $|f|$ and $r_{\text{norm}}$ show different rates; $|f|$ needs less iteration steps than $r_{\text{norm}}$; The convergence rate of $r_{\text{norm}}$ lags behind that of $|f|$. It should be mentioned here for this complicated Dafalias-Manzari model considering fabric dilation effect,
the typical constitutive level convergence performance is similar to that of Drucker-Prager model with Armstrong-Frederick kinematic hardening, but with much smaller strain increment interval.

From the above examples, it is clear that the simpler the model is, the better constitutive level convergence performances are observed. This is consistent to the error assessment in section 304.2.1. Generally, the implemented implicit algorithm shows stable constitutive level convergence performances provided an appropriate small strain increment interval for the material model.

304.3 Validation of Constitutive Model Predictions

304.3.1 Dafalias Manzari Material Model

Validation is performed by comparing experimental (physical) results and numerical (constitutive) simulations for the Toyoura sand. It should be noted that we have not done validation against 2D or 3D tests (say centrifuge tests) as characterization of sand used in centrifuge experiments is usually less than complete for use with advanced constitutive models. Moreover, as our approach seeks to make predictions of prototype behavior, scaling down models (and using them for comparison with numerical predictions) brings forward issues of physics of scaling which we would rather stay out of. The material parameters
Figure 304.15: Typical convergence for Dafalias-Manzari model (tolerance value $1 \times 10^{-7}$).

used are from Dafalias and Manzari (2004a) and are listed in Table (304.1). Several simulated results are compared with the experimental data published by Verdugo and Ishihara (1996).

Figure (304.16) presents both loading and unloading triaxial drained test simulation results for a relatively low triaxial isotropic pressure of 100 kPa but with different void ratios of $e_0 = 0.831, 0.917, 0.996$ at the end of isotropic compression. During the loading stage, one can observe the hardening and then softening together with the contraction and then dilation for the denser sand, while only hardening together with contraction for the looser sand. The significance of the state parameter to the soil modeling is clear from the very different responses with different void ratios at the same triaxial isotropic pressure.

Figure (304.16) also shows both loading and unloading triaxial drained test simulation results for a relatively high triaxial isotropic pressure of 500 kPa but with different void ratio of $e_0 = 0.810, 0.886, 0.960$ at the end of isotropic compression. Similar phenomenon are observed as with tests (physical and numerical) for relatively low triaxial isotropic pressure. However, due to the higher confinement pressure, the stress-strain responses are higher at the same strain, which proves the significant pressure dependent feature for the sands.

Figure (304.17) presents both loading and unloading triaxial undrained test simulation results for a dense sand with the void ratio of $e_0 = 0.735$ at the end of isotropic compression but with different isotropic compression pressures of $p_0 = 100, 1000, 2000, 3000$ kPa. During the loading stage, one
Figure 304.16: Left: Experimental data; Right: Simulated results.
Figure 304.17: Left: Experimental data; Right: Simulated results.
observes that each of responses are close to the critical state line for the very various range of isotropic compression pressures. For the higher isotropic compression pressure, the contraction response with softening is clearly observed, while for the smaller isotropic compression pressure, it is a dilation response without softening.

Close matching of physical testing data with constitutive predictions represents a satisfactory validation of our material model. This validation with previous verification gives us confidence that predictions (presented in next section) represent well the real, prototype behavior.
Chapter 305

Verification and Validation for Static and Dynamic Finite Element Level Solution Advancement Algorithms


(In collaboration with Prof. José Antonio Abell Mena and Dr. Yuan Feng)
305.1 Chapter Summary and Highlights

305.2 Verification for Static Solution Advancement

305.3 Verification for Dynamic Solution Advancement

Spectral Radii, Argyris and Mlejnek (1991)...

- Test consists of a 2-DOF system periods $T_1 = 4.0 \, \text{s}$ and $T_2 = 1.0 \, \text{s}$.
- Different integration steps.
- Different values of method parameters ($\alpha$, $\beta$, $\gamma$).
- Results are compared with theoretical predictions for algorithmic damping and period shifts.

305.3.1 Verification for Dynamic Solution Advancement, Newmark Method

(Newmark, 1959)

If

$$\gamma \geq \frac{1}{2}, \quad \beta \geq \frac{1}{4}(\gamma + \frac{1}{2})^2$$

(305.1)

the procedure is unconditionally stable and second-order accurate.

Different values for $\gamma$ and $\beta$ can be used to create various integration methods:

- For $\gamma = 0.5$ (and corresponding $\beta = 0.25$) there is no numerical damping.
- Any $\gamma$ value greater than 0.5 will introduce numerical damping.
- Trapezoidal rule or average acceleration method for $\beta = 1/4$ and $\gamma = 1/2$,
- Linear acceleration method for $\beta = 1/6$ and $\gamma = 1/2$,
- Explicit, central difference method for $\beta = 0$ and $\gamma = 1/2$.
- Strongest numerical damping values is obtained for values $\beta = 1$ and $\gamma = 2/3$, as spectral ratio $\rho_{\infty} = 0$ (Hughes (1987), page 502)

For more details see chapter 108 on page 506.
Following Argyris and Mlejnek (1991); Hughes (1987), to calculate the analytic $\xi$ and analytic $\bar{\omega}$ a matrix $A$ is constructed. The explicit definition of amplification matrix $A$ for the Hilber Hughes Taylor (HHT) family of algorithms (where Newmark is obtained by setting $\gamma = 0$) is

$$A = \frac{1}{D} \begin{bmatrix} 1 + \alpha \beta \Omega^2 & 1 & \frac{1}{2} - \beta \\ -\gamma \Omega^2 & 1 - (1 + \alpha)(\gamma - \beta) \Omega^2 & 1 - \gamma - (1 + \alpha)(\frac{1}{2} \gamma - \beta) \Omega^2 \\ -\Omega^2 & -(1 + \alpha) \Omega^2 & -(1 + \alpha)(\frac{1}{2} - \beta) \Omega^2 \end{bmatrix} \quad (305.2)$$

where

$$D = 1 + (1 + \alpha) \beta \Omega^2$$

$$\Omega = \omega \Delta t \quad (305.3)$$

$$\omega = (K/M)^{\frac{1}{2}}$$

The eigenvalue of amplification matrix $A$ will be two complex conjugate roots $\lambda_{1,2}$ and a so-called spurious root $\lambda_3$ which satisfy $|\lambda_3| < |\lambda_{1,2}| \leq 1$. The roots $\lambda_{1,2}$ will be

$$\lambda_{1,2} = A \pm Bi \quad (305.4)$$

Then, the analytic damping ratio $\xi$ and analytic period $\bar{\omega}$ becomes

$$\bar{\xi} = -\ln(A^2 + B^2)$$

$$\bar{\omega} = \Omega / \Delta t$$

$$\bar{\Omega} = \arctan(B/A) \quad (305.5)$$

<table>
<thead>
<tr>
<th>dt</th>
<th>gamma damping</th>
<th>Th. damp.</th>
<th>T shift</th>
<th>Th. T. shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0100 1.5000 3.398552 3.139529 0.00000000 -0.13145647</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.0050 0.5000 0.295322 -0.000000 0.00000000 -0.00822413</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.0500 0.5000 0.368976 0.000000 -1.01010101 -0.81712426</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.1250 0.5000 0.027518 0.000000 -4.95382032 -4.9456802</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.0050 0.5100 0.298179 0.015705 0.00000000 -0.00822660</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.0500 0.5100 0.395657 0.154550 -1.01010101 -0.81736312</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.1250 0.5100 0.346769 0.157054 0.00000000 -0.00847079</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.0050 0.6000 0.346769 0.157054 0.00000000 -0.00847079</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.0500 0.6000 1.600134 1.545504 -1.01010101 -0.84101009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.1250 0.6000 3.571434 3.570958 -5.04201681 -5.07208349</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.0050 0.7000 0.418286 0.314108 0.00000000 -0.00921077</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.0500 0.7000 3.113067 3.091044 -1.01010101 -0.91266978</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.1250 0.7000 7.130075 7.144404 -5.13036165 -5.45499548</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.0050 0.9000 0.625858 0.628215 0.00000000 -0.01217067</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.0500 0.9000 6.177709 6.182372 -1.01010101 -1.19934156</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
305.3.2 Verification Example Description.

A one degree of freedom (DOF) example was made to verify the Newmark and HHT algorithm for Real-ESSI simulator. The example was plot below in Fig. (305.7). The beam stiffness and the mass were designed to make the natural period to be 1 second. In the first loading stage, the beam was given a horizontal force to generate an initial displacement. By the way, at the top node, all DOFs were fixed except the DOF along initial displacement. Then, in the second loading stage, the beam start free vibration.

The results were listed and plotted below.
Figure 305.2: Verification: Dynamic solution advancement, Newmark method (p2a-newmark-05errors).

Figure 305.3: Verification: Dynamic solution advancement, Newmark method (p2a-newmark-06top).
Figure 305.4: Verification: Dynamic solution advancement, Newmark method (p2a-newmark-05errors)

Figure 305.5: Verification: Dynamic solution advancement, Newmark method (p2a-newmark-08top)
Figure 305.6: Verification: Dynamic solution advancement, Newmark method (p2a-newmark-08errors).

Figure 305.7: Verification example description.
Table 305.1: Verification results for the Newmark solution advancement algorithm.

<table>
<thead>
<tr>
<th>$dt$</th>
<th>$\gamma$</th>
<th>Measured $\xi$</th>
<th>Analytic $\xi$</th>
<th>Measured $T$ shift</th>
<th>Analytic $T$ shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>0.5</td>
<td>0.0030</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.01</td>
<td>0.5</td>
<td>0.0030</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0003</td>
</tr>
<tr>
<td>0.05</td>
<td>0.5</td>
<td>0.0037</td>
<td>0.0000</td>
<td>0.0101</td>
<td>0.0082</td>
</tr>
<tr>
<td>0.1</td>
<td>0.5</td>
<td>0.0034</td>
<td>0.0000</td>
<td>0.0309</td>
<td>0.0321</td>
</tr>
<tr>
<td>0.005</td>
<td>0.6</td>
<td>0.0035</td>
<td>0.0016</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.01</td>
<td>0.6</td>
<td>0.0042</td>
<td>0.0031</td>
<td>0.0000</td>
<td>0.0003</td>
</tr>
<tr>
<td>0.05</td>
<td>0.6</td>
<td>0.0160</td>
<td>0.0155</td>
<td>0.0101</td>
<td>0.0084</td>
</tr>
<tr>
<td>0.1</td>
<td>0.6</td>
<td>0.0296</td>
<td>0.0295</td>
<td>0.0309</td>
<td>0.0329</td>
</tr>
<tr>
<td>0.005</td>
<td>0.7</td>
<td>0.0042</td>
<td>0.0031</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.01</td>
<td>0.7</td>
<td>0.0063</td>
<td>0.0063</td>
<td>0.0000</td>
<td>0.0004</td>
</tr>
<tr>
<td>0.05</td>
<td>0.7</td>
<td>0.0311</td>
<td>0.0309</td>
<td>0.0101</td>
<td>0.0091</td>
</tr>
<tr>
<td>0.1</td>
<td>0.7</td>
<td>0.0590</td>
<td>0.0590</td>
<td>0.0309</td>
<td>0.0356</td>
</tr>
<tr>
<td>0.005</td>
<td>0.8</td>
<td>0.0051</td>
<td>0.0047</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.01</td>
<td>0.8</td>
<td>0.0093</td>
<td>0.0094</td>
<td>0.0000</td>
<td>0.0004</td>
</tr>
<tr>
<td>0.05</td>
<td>0.8</td>
<td>0.0465</td>
<td>0.0464</td>
<td>0.0101</td>
<td>0.0103</td>
</tr>
<tr>
<td>0.1</td>
<td>0.8</td>
<td>0.0882</td>
<td>0.0886</td>
<td>0.0309</td>
<td>0.0399</td>
</tr>
<tr>
<td>0.005</td>
<td>0.9</td>
<td>0.0063</td>
<td>0.0063</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>0.01</td>
<td>0.9</td>
<td>0.0130</td>
<td>0.0126</td>
<td>0.0000</td>
<td>0.0005</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9</td>
<td>0.0618</td>
<td>0.0618</td>
<td>0.0101</td>
<td>0.0120</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>0.1180</td>
<td>0.1181</td>
<td>0.0417</td>
<td>0.0460</td>
</tr>
</tbody>
</table>

Verification Results for Newmark Solution Advancement Algorithm. The Real-ESSI model fei/DSL files for the results above can be downloaded [HERE](#).
Figure 305.8: Comparison for Newmark algorithm with $\gamma = 0.5$. Damping ratio comparison, Period shift comparison.

Figure 305.9: Comparison for Newmark algorithm with $\gamma = 0.6$. Damping ratio comparison, Period shift comparison.

Figure 305.10: Comparison for Newmark algorithm with $\gamma = 0.7$. Damping ratio comparison, Period shift comparison.
Figure 305.11: Comparison for Newmark algorithm with $\gamma = 0.8$. Damping ratio comparison, Period shift comparison.

Figure 305.12: Comparison for Newmark algorithm with $\gamma = 0.9$. Damping ratio comparison, Period shift comparison.
305.3.3 Verification for Dynamic Solution Advancement, Hilber-Hughes-Taylor Method

(Hilber et al., 1977), (Hughes and Liu, 1978a) and (Hughes and Liu, 1978b)

If the parameters $\alpha$, $\beta$ and $\gamma$ satisfy

$$-\frac{1}{3} \leq \alpha \leq 0, \quad \gamma = \frac{1}{2} (1 - 2\alpha), \quad \beta = \frac{1}{4} (1 - \alpha)^2 \tag{305.6}$$

it is unconditionally stable and second-order accurate (Argyris and Mlejnek, 1991; Hughes, 1987).

![Figure 305.13: Verification: Dynamic solution advancement, Hilber-Hughes-Taylor method (p2a-hht-01top)...](image)

**Summary**

<table>
<thead>
<tr>
<th>$dt$</th>
<th>$\alpha$ damping [%]</th>
<th>Th. damp. [%]</th>
<th>T shift [%]</th>
<th>Th. T. shift [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0050</td>
<td>-0.0000</td>
<td>0.295322</td>
<td>-0.000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0.0100</td>
<td>-0.0000</td>
<td>0.302864</td>
<td>0.000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0.0500</td>
<td>-0.0000</td>
<td>0.368976</td>
<td>-1.01010101</td>
<td>-0.81712426</td>
</tr>
<tr>
<td>0.1000</td>
<td>-0.0000</td>
<td>0.390004</td>
<td>-3.09278351</td>
<td>-3.20749106</td>
</tr>
<tr>
<td>0.2500</td>
<td>-0.0000</td>
<td>0.458456</td>
<td>-17.64705882</td>
<td>-17.96772753</td>
</tr>
<tr>
<td>0.0050</td>
<td>-0.0100</td>
<td>0.295394</td>
<td>0.000004</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0.0100</td>
<td>-0.0100</td>
<td>0.303170</td>
<td>0.000030</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0.0500</td>
<td>-0.0100</td>
<td>0.357316</td>
<td>-1.01010101</td>
<td>-0.84065236</td>
</tr>
<tr>
<td>0.1000</td>
<td>-0.0100</td>
<td>0.376004</td>
<td>-3.09278351</td>
<td>-3.29331088</td>
</tr>
<tr>
<td>0.2500</td>
<td>-0.0100</td>
<td>0.458456</td>
<td>-17.64705882</td>
<td>-18.29362011</td>
</tr>
<tr>
<td>0.0050</td>
<td>-0.1000</td>
<td>0.295948</td>
<td>0.000031</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0.0100</td>
<td>-0.1000</td>
<td>0.303170</td>
<td>0.000030</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0.0500</td>
<td>-0.1000</td>
<td>0.357316</td>
<td>-1.01010101</td>
<td>-0.84065236</td>
</tr>
<tr>
<td>0.1000</td>
<td>-0.1000</td>
<td>0.376004</td>
<td>-3.09278351</td>
<td>-3.29331088</td>
</tr>
<tr>
<td>0.2500</td>
<td>-0.1000</td>
<td>0.458456</td>
<td>-17.64705882</td>
<td>-18.29362011</td>
</tr>
<tr>
<td>0.0050</td>
<td>-0.2000</td>
<td>0.295948</td>
<td>0.000031</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0.0100</td>
<td>-0.2000</td>
<td>0.303170</td>
<td>0.000030</td>
<td>0.00000000</td>
</tr>
</tbody>
</table>
### 305.3.3.1 Verification Results for Hilber Hughes Taylor (HHT) Solution Advancement Algorithm.

The Real-ESSI model fei/DSL files for the results above can be downloaded [HERE](#).
Table 305.2: Verification results for Hilber Hughes Taylor (HHT) solution advancement algorithm.

<table>
<thead>
<tr>
<th>dt</th>
<th>α</th>
<th>Measured ξ</th>
<th>Analytic ξ</th>
<th>Measured T shift</th>
<th>Analytic T shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>-0.0</td>
<td>0.002953</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000082</td>
</tr>
<tr>
<td>0.01</td>
<td>-0.0</td>
<td>0.003029</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000329</td>
</tr>
<tr>
<td>0.05</td>
<td>-0.0</td>
<td>0.003690</td>
<td>0.000000</td>
<td>0.010101</td>
<td>0.008171</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.0</td>
<td>0.003390</td>
<td>0.000000</td>
<td>0.030928</td>
<td>0.032075</td>
</tr>
<tr>
<td>0.005</td>
<td>-0.01</td>
<td>0.002954</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000085</td>
</tr>
<tr>
<td>0.01</td>
<td>-0.01</td>
<td>0.003032</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000339</td>
</tr>
<tr>
<td>0.05</td>
<td>-0.01</td>
<td>0.003573</td>
<td>0.000036</td>
<td>0.010101</td>
<td>0.008407</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.01</td>
<td>0.003768</td>
<td>0.000259</td>
<td>0.030928</td>
<td>0.032933</td>
</tr>
<tr>
<td>0.005</td>
<td>-0.05</td>
<td>0.002957</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000094</td>
</tr>
<tr>
<td>0.01</td>
<td>-0.05</td>
<td>0.003043</td>
<td>0.000001</td>
<td>0.000000</td>
<td>0.000374</td>
</tr>
<tr>
<td>0.05</td>
<td>-0.05</td>
<td>0.003227</td>
<td>0.000167</td>
<td>0.010101</td>
<td>0.009276</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.05</td>
<td>0.00382</td>
<td>0.001184</td>
<td>0.030928</td>
<td>0.036111</td>
</tr>
<tr>
<td>0.005</td>
<td>-0.1</td>
<td>0.002959</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000103</td>
</tr>
<tr>
<td>0.01</td>
<td>-0.1</td>
<td>0.003055</td>
<td>0.000003</td>
<td>0.000000</td>
<td>0.000413</td>
</tr>
<tr>
<td>0.05</td>
<td>-0.1</td>
<td>0.003054</td>
<td>0.000300</td>
<td>0.010101</td>
<td>0.010202</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.1</td>
<td>0.004321</td>
<td>0.002106</td>
<td>0.041667</td>
<td>0.039506</td>
</tr>
<tr>
<td>0.005</td>
<td>-0.2</td>
<td>0.002964</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000117</td>
</tr>
<tr>
<td>0.01</td>
<td>-0.2</td>
<td>0.003073</td>
<td>0.000004</td>
<td>0.000000</td>
<td>0.000467</td>
</tr>
<tr>
<td>0.05</td>
<td>-0.2</td>
<td>0.003479</td>
<td>0.000472</td>
<td>0.010101</td>
<td>0.011518</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.2</td>
<td>0.005194</td>
<td>0.003286</td>
<td>0.041667</td>
<td>0.044350</td>
</tr>
</tbody>
</table>
Figure 305.15: Verification: Dynamic solution advancement, Hilber-Hughes-Taylor method (p2a-hht-02top)
Figure 305.16: Verification: Dynamic solution advancement, Hilber-Hughes-Taylor method (p2a-hht-01errors).

Figure 305.17: Comparison for HHT algorithm with $\alpha = -0.0$. Damping ratio comparison, Period shift comparison.
Figure 305.18: Comparison for HHT algorithm with $\alpha = -0.01$. Damping ratio comparison, Period shift comparison.

Figure 305.19: Comparison for HHT algorithm with $\alpha = -0.05$. Damping ratio comparison, Period shift comparison.

Figure 305.20: Comparison for HHT algorithm with $\alpha = -0.10$. Damping ratio comparison, Period shift comparison.
Figure 305.21: Comparison for HHT algorithm with $\alpha = -0.20$. Damping ratio comparison, Period shift comparison.
Chapter 306

Verification and Validation for Static and Dynamic Behavior of Single Phase, Solid Elements


(In collaboration with Dr. Yuan Feng, and Dr. Han Yang)
306.1 Chapter Summary and Highlights

306.2 Verification of Static, Single Phase Solid Modeling and Simulation

306.2.1 Beam theory

This section provides basic beam theory that is used for verification solutions.

Problem description: Length=6m, Width=1m, Height=1m, F=100N, E=1E8Pa, \( \nu = 0.0 \). The force direction was shown in Figure (306.1).

![Figure 306.1: Problem description for cantilever beams.](image)

The basic idea to calculate the shear deformation of a beam is

\[
\delta = \frac{FL}{GA_v} \tag{306.1}
\]

where \( A_v \) is the not the gross cross sectional area of the beam. \( A_v \) should be the shear area. Thus,

\[
\kappa = \frac{A}{A_v} \tag{306.2}
\]

where \( \kappa \) is the form factor, shear correction factor or shear deformation coefficient, \( A \) is the gross sectional area and \( A_v \) is the shear area of the section.

REWRITE, use bibtex! The history of \( \kappa \) value is long.

1. Timoshenko (1940)\(^1\) define the form factor for rectangular section is 1.5.

\(^1\)Strength of materials, Timoshenko, Krieger Pub Co, 1940
2. Cowper (1970)² gave the formula for the form factor:

\[ \kappa = \frac{12 + 11\nu}{10(1+\nu)} \]  

(306.3)

3. Renton (1991)³ provided a closed form solution for shear area of rectangular sections. For a rectangular section of depth \(2a\) and breadth \(2b\),

\[ \kappa = \frac{6}{5} + \left(\frac{\nu}{1+\nu}\right)^2 \sum_{m=0}^{\infty} \sum_{n=1}^{\infty} \frac{144(b/a)^4}{\pi^6(2m+1)^2 n^2[(2m+1)^2(b/2a)^2 + n^2]} \]  

(306.4)

For square cross section, \(b = a\), therefore,

\[ \kappa = \frac{6}{5} + \left(\frac{\nu}{1+\nu}\right)^2 \sum_{m=0}^{\infty} \sum_{n=1}^{\infty} \frac{144}{\pi^6(2m+1)^2 n^2[(2m+1)^2(1/2)^2 + n^2]} \]  

(306.5)

To simplify the equation above, according to the Renton (1991), the intermediate values are given by

\[ \kappa = \frac{6}{5} + C_1 \left(\frac{\nu}{1+\nu}\right)^2 \left(\frac{b}{a}\right)^4 \]  

(306.6)

where \(C_1\) is the parameter determined by the ratio of \(a\) and \(b\). When \(b = a\), the equation becomes

\[ \kappa = \frac{6}{5} + 0.1392 \left(\frac{\nu}{1+\nu}\right)^2 \]  

(306.7)

306.2.2 Verification of 8 node brick cantilever beam (static)

Problem description: Lenght=10m, Force=4N, E=100000Pa, \(I = \frac{1}{12}\)

Theoretical displacement:

\[ d = \frac{PL^3}{3EI} = \frac{4 \times 1000}{3 \times 100000 \times \frac{1}{12}} = 0.16m \]  

(306.8)

Numerical simulation results:

1m element size (10 elements):

\[ error = \frac{0.16 - 0.1072}{0.16} = 33\% \] (306.9)

5m element size (2 elements):

\[ error = \frac{0.16 - 0.011911}{0.16} = 92.5\% \] (306.10)

10m element size (1 element):

\[ error = \frac{0.16 - 0.00315}{0.16} = 98\% \] (306.11)

1m element size with 10% nodal offset (10 elements):

\[ error = \frac{0.16 - 0.1057}{0.16} = 34\% \] (306.12)

1m element size with 20% nodal offset (10 elements):

\[ error = \frac{0.16 - 0.1016}{0.16} = 36\% \] (306.13)
306.2.3 Verification of 27 node brick cantilever beam (static)

Problem description: Lenght=10m, Force=9N, E=100000Pa, $I = \frac{1}{12}$

Theoretical displacement:

$$d = \frac{PL^3}{3EI} = \frac{9 \times 1000}{3 \times 100000 \times \frac{1}{12}} = 0.36m \quad (306.14)$$

Numerical simulation results:

1m element size (10 elements):

Error

$$error = \frac{0.361721 - 0.36}{0.36} = 0.47\% \quad (306.15)$$

5m element size (2 elements):

Error

$$error = \frac{0.36 - 0.345719}{0.36} = 3.96\% \quad (306.16)$$

10m element size (1 element):
\[
error = \frac{0.36 - 0.279989}{0.36} = 22\%
\]  
(306.17)

1m element size with 10\% nodal offset (10 elements):

\[
error = \frac{0.361225 - 0.36}{0.36} = 0.35\%
\]  
(306.18)

1m element size with 20\% nodal offset (10 elements):

\[
error = \frac{0.36 - 0.359741}{0.36} = 0.07\%
\]  
(306.19)

1m element size with 30\% nodal offset (10 elements):

\[
error = \frac{0.36 - 0.357004}{0.36} = 0.83\%
\]  
(306.20)

1m element size with 40\% nodal offset (10 elements):

\[
error = \frac{0.36 - 0.352604}{0.36} = 2\%
\]  
(306.21)
306.2.4 Verification of 8NodeBrick cantilever beams

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \). Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (306.2).

\[ \begin{align*}
\text{Theoretical displacement (bending and shear deformation):} \\
\quad d & = \frac{FL^3}{3EI} + \frac{FL}{GA_v} \\
\quad & = \frac{FL^3}{3E \frac{bh^3}{12}} + \frac{FL}{\frac{E}{2(1+\nu)} \frac{bh}{\kappa}} \\
\quad & = \frac{100N \times 6^3 m^3}{3 \times 10^8 N/m^2 \times \frac{1}{12} m^4} + \frac{100N \times 6m}{\frac{10}{2} \times 10^7 N/m^2 \times 1m^2 \times \frac{5}{6}} \\
\quad & = 8.64 \times 10^{-4} m + 0.144 \times 10^{-4} m \\
\quad & = 8.784 \times 10^{-4} m
\end{align*} \]  

(306.22)

Numerical model:

The 8NodeBrick elements are shown in Figure (306.6).

An example Real-ESSI script is shown below.

All the Real-ESSI results are listed in Table (306.1). The theoretical solution is 8.784E-04 m.

<table>
<thead>
<tr>
<th>Element number</th>
<th>1</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8NodeBrick</td>
<td>4.61E-05 m</td>
<td>1.59E-04 m</td>
<td>5.84E-04 m</td>
</tr>
<tr>
<td>Error</td>
<td>94.75%</td>
<td>81.87%</td>
<td>33.52%</td>
</tr>
</tbody>
</table>

The errors are plotted in Figure (306.7).

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.3: One 8NodeBrick element.

Figure 306.4: Two 8NodeBrick elements.

Figure 306.5: Six 8NodeBrick elements.

Figure 306.6: 8NodeBrick elements for cantilever beams.

Figure 306.7: 8NodeBrick cantilever beam for different element number Displacement error versus Number of elements
306.2.5 Verification of 8NodeBrick cantilever beam for different Poisson’s ratio

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.0 - 0.49$. The force direction was shown in Figure (306.8).

![Figure 306.8: Problem description for cantilever beams of different Poisson's ratios.](image)

The theoretical solution for $\nu = 0.0$ was calculated below, while the solution for other Poisson’s ratio are calculated by the similar process.

Theoretical displacement (bending and shear deformation):

\[
\begin{align*}
  d &= \frac{FL^3}{3EI} + \frac{FL}{GA_v} \\
  &= \frac{FL^3}{3EI} + \frac{FL}{2(1+\nu) \frac{bh}{12}} \\
  &= \frac{100N \times 6^3m^3}{3 \times 10^8N/m^2 \times \frac{1}{12}m^4} + \frac{100N \times 6m}{\frac{10}{2} \times 10^2N/m^2 \times 1m^2 \times \frac{5}{6}} \\
  &= 8.64 \times 10^{-4}m + 0.144 \times 10^{-4}m \\
  &= 8.784 \times 10^{-4} m 
\end{align*}
\]  

(306.23)

The rotation angle at the end:

\[
\begin{align*}
  \theta &= \frac{FL^2}{2EI} = \frac{100N \times 6^2m^2}{2 \times 10^8N/m^2 \times \frac{1}{12}m^4} = 2.16 \times 10^{-4} rad = 0.0124^\circ 
\end{align*}
\]  

(306.24)

The 8NodeBrick elements for cantilever beams of different Poisson’s ratios are shown in Figure (306.9):

All the displacement results are listed in Table (306.2) - (306.4).

Using the same geometry, the element was meshed using much smaller element (0.5m).

Finally, in the same geometry, the element side length was cut into 0.25m.

The errors are plotted in Figure (306.10).
Figure 306.9: 8NodeBrick elements for cantilever beams of different Poisson’s ratios.

Table 306.2: **Displacement** results for 8NodeBrick cantilever beams with element side length 1 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>8NodeBrick displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement (all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>5.840E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
</tr>
<tr>
<td>0.05</td>
<td>5.924E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
</tr>
<tr>
<td>0.10</td>
<td>5.969E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
</tr>
<tr>
<td>0.15</td>
<td>5.971E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
</tr>
<tr>
<td>0.20</td>
<td>5.922E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
</tr>
<tr>
<td>0.25</td>
<td>5.814E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
</tr>
<tr>
<td>0.30</td>
<td>5.634E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
</tr>
<tr>
<td>0.35</td>
<td>5.364E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
</tr>
<tr>
<td>0.40</td>
<td>4.970E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
</tr>
<tr>
<td>0.45</td>
<td>4.353E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
</tr>
<tr>
<td>0.49</td>
<td>3.142E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
</tr>
</tbody>
</table>

Figure 306.10: 8NodeBrick cantilever beam for different Poisson’s ratio. **Displacement error versus Poisson’s ratio.** Left: Error scale 0% - 80%, Right: Error scale 0% - 100%.
Table 306.3: **Displacement** results for 8NodeBrick cantilever beams with element side length 0.5 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>8NodeBrick displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>7.787E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>11.35%</td>
</tr>
<tr>
<td>0.05</td>
<td>7.824E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>11.00%</td>
</tr>
<tr>
<td>0.10</td>
<td>7.839E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>10.91%</td>
</tr>
<tr>
<td>0.15</td>
<td>7.829E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>11.09%</td>
</tr>
<tr>
<td>0.20</td>
<td>7.790E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>11.61%</td>
</tr>
<tr>
<td>0.25</td>
<td>7.717E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>12.51%</td>
</tr>
<tr>
<td>0.30</td>
<td>7.597E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>13.95%</td>
</tr>
<tr>
<td>0.35</td>
<td>7.406E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>16.18%</td>
</tr>
<tr>
<td>0.40</td>
<td>7.089E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>19.84%</td>
</tr>
<tr>
<td>0.45</td>
<td>6.466E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>26.95%</td>
</tr>
<tr>
<td>0.49</td>
<td>4.990E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>43.66%</td>
</tr>
</tbody>
</table>

Table 306.4: **Displacement** results for 8NodeBrick cantilever beams with element side length 0.25 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>8NodeBrick displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.511E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>3.11%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.525E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>3.03%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.527E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>3.09%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.518E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>3.27%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.494E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>3.62%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.455E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>4.15%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.393E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>4.93%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.299E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>6.08%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.141E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>7.94%</td>
</tr>
<tr>
<td>0.45</td>
<td>7.801E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>11.86%</td>
</tr>
<tr>
<td>0.49</td>
<td>6.603E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>25.45%</td>
</tr>
</tbody>
</table>

The angle results are listed in Table (306.5).

Then, in the same geometry, element side length was cut into 0.5m. The angle results are listed in Table (306.6).
### Table 306.5: Rotation angle results for 8NodeBrick cantilever beams with element side length 1 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>8NodeBrick angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.25E-03</td>
<td>1.24E-02</td>
<td>33.46%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.36E-03</td>
<td>1.24E-02</td>
<td>32.55%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.42E-03</td>
<td>1.24E-02</td>
<td>32.08%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.42E-03</td>
<td>1.24E-02</td>
<td>32.10%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.35E-03</td>
<td>1.24E-02</td>
<td>32.67%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.20E-03</td>
<td>1.24E-02</td>
<td>33.90%</td>
</tr>
<tr>
<td>0.30</td>
<td>7.95E-03</td>
<td>1.24E-02</td>
<td>35.89%</td>
</tr>
<tr>
<td>0.35</td>
<td>7.59E-03</td>
<td>1.24E-02</td>
<td>38.83%</td>
</tr>
<tr>
<td>0.40</td>
<td>7.07E-03</td>
<td>1.24E-02</td>
<td>43.00%</td>
</tr>
<tr>
<td>0.45</td>
<td>6.30E-03</td>
<td>1.24E-02</td>
<td>49.21%</td>
</tr>
<tr>
<td>0.49</td>
<td>4.93E-03</td>
<td>1.24E-02</td>
<td>60.20%</td>
</tr>
</tbody>
</table>

### Table 306.6: Rotation angle results for 8NodeBrick cantilever beams with element side length 0.5 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>8NodeBrick angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.10E-02</td>
<td>1.24E-02</td>
<td>11.28%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.10E-02</td>
<td>1.24E-02</td>
<td>10.91%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.11E-02</td>
<td>1.24E-02</td>
<td>10.78%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.10E-02</td>
<td>1.24E-02</td>
<td>10.90%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.10E-02</td>
<td>1.24E-02</td>
<td>11.32%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.09E-02</td>
<td>1.24E-02</td>
<td>12.09%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.07E-02</td>
<td>1.24E-02</td>
<td>13.33%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.05E-02</td>
<td>1.24E-02</td>
<td>15.29%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.01E-02</td>
<td>1.24E-02</td>
<td>18.53%</td>
</tr>
<tr>
<td>0.45</td>
<td>9.32E-03</td>
<td>1.24E-02</td>
<td>24.87%</td>
</tr>
<tr>
<td>0.49</td>
<td>7.52E-03</td>
<td>1.24E-02</td>
<td>39.35%</td>
</tr>
</tbody>
</table>

Finally, in the same geometry, element side length was cut into 0.25m. The angle results are listed in Table (306.7).
Table 306.7: **Rotation angle** results for 8NodeBrick cantilever beams with element side length 0.25 m.

<table>
<thead>
<tr>
<th>Poisson's ratio</th>
<th>8NodeBrick angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.20E-02</td>
<td>1.24E-02</td>
<td>3.06%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.20E-02</td>
<td>1.24E-02</td>
<td>2.97%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.20E-02</td>
<td>1.24E-02</td>
<td>2.99%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.20E-02</td>
<td>1.24E-02</td>
<td>3.12%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.20E-02</td>
<td>1.24E-02</td>
<td>3.38%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.19E-02</td>
<td>1.24E-02</td>
<td>3.79%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.19E-02</td>
<td>1.24E-02</td>
<td>4.40%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.17E-02</td>
<td>1.24E-02</td>
<td>5.33%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.15E-02</td>
<td>1.24E-02</td>
<td>6.87%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.11E-02</td>
<td>1.24E-02</td>
<td>10.22%</td>
</tr>
<tr>
<td>0.49</td>
<td>9.64E-03</td>
<td>1.24E-02</td>
<td>22.23%</td>
</tr>
</tbody>
</table>

The errors are plotted in Figure (306.11).

Figure 306.11: 8NodeBrick cantilever beam for different Poisson’s ratio Rotation angle error versus Poisson’s ratio, Left: Error scale 30% - 70%, Right: Error scale 0% - 100%.

The Real-ESSI model fei/DSL files for the table above are HERE.
306.2.6 Test of irregular shaped 8NodeBrick cantilever beams

Cantilever model was used as an example. Three different shapes are tested.

In the first test, the upper two nodes of each element are moved one half element size along the \( y \) \(-\)axis, while the lower two nodes are kept at the same location. The element shape was shown in Figure (306.12).

![Figure 306.12: 8NodeBrick cantilever beams for irregular Shape 1.](image)

In the second test, the upper two nodes of each element are moved 90\% element size along the \( y \) \(-\)axis, while the lower two nodes are moved 90\% element size in the other direction along the \( y \) \(-\)axis. The element shape was shown in Figure (306.13).

![Figure 306.13: 8NodeBrick cantilever beams for irregular Shape 2.](image)

In the third test, the upper two nodes of each element are moved one half element size with different directions along the \( y \) \(-\)axis, while the lower two nodes are kept at the same location. The element shape was shown in Figure (306.14).

![Figure 306.14: 8NodeBrick cantilever beams for irregular Shape 3.](image)

The boundary conditions are shown in Figures (306.17), (306.20) and (306.23).

The Real-ESSI results are listed in Table (306.8).

The errors are listed in Table (306.9) and (306.10).

The Real-ESSI model fei/DSL files for the table above are HERE.

Then, the irregular beam was divided into small elements.

Problem description: Length=12m, Width=2m, Height=2m, \( q=400\text{N/m} \), \( E=1\text{E}8\text{Pa} \), \( \nu=0.0 \). Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (306.24).
Figure 306.14: 8NodeBrick cantilever beams for irregular Shape 3.

Figure 306.15: Vertical force.

Figure 306.16: Horizontal force.

Figure 306.17: 8NodeBrick cantilever beam boundary conditions for irregular Shape 1.

Table 306.8: Results for 8NodeBrick cantilever beams of irregular shapes.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Force direction</th>
<th>Normal shape</th>
<th>Shape 1</th>
<th>Shape 2</th>
<th>Shape 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8NodeBrick</td>
<td>Vertical (z)</td>
<td>5.840E-04 m</td>
<td>5.751E-04 m</td>
<td>2.959E-04 m</td>
<td>3.883E-04 m</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>Transverse (y)</td>
<td>5.840E-04 m</td>
<td>4.529E-04 m</td>
<td>1.390E-04 m</td>
<td>4.744E-04 m</td>
</tr>
<tr>
<td>Theoretical</td>
<td>-</td>
<td>8.784E-04 m</td>
<td>8.784E-04 m</td>
<td>8.784E-04 m</td>
<td>8.784E-04 m</td>
</tr>
</tbody>
</table>

Table 306.9: Errors for irregular shaped 8NodeBrick compared to theoretical solution.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Force direction</th>
<th>Normal shape</th>
<th>Shape 1</th>
<th>Shape 2</th>
<th>Shape 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8NodeBrick</td>
<td>Vertical (z)</td>
<td>33.52%</td>
<td>34.53%</td>
<td>66.31%</td>
<td>55.79%</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>Transverse (y)</td>
<td>33.52%</td>
<td>48.44%</td>
<td>84.18%</td>
<td>45.99%</td>
</tr>
</tbody>
</table>
Theoretical displacement (bending and shear deformation):

\[ d = \frac{qL^4}{8EI} + \frac{qL^2}{2CA} \]  

(306.25)

\[ d = \frac{qL^4}{8EHh^2} + \frac{qL^2}{2(1+\nu)bh} \]  

(306.26)

\[ d = \frac{400 \text{N/m} \times 12^4 \text{m}^4}{8 \times 10^8 \text{N/m}^2 \times \frac{2^4}{12} \text{m}^3} + \frac{400 \text{N/m} \times 12^2 \text{m}^2}{\frac{10^8}{2} \text{N/m}^2 \times 2m \times 2m \times \frac{5}{6}} \]

\[ = 7.776 \times 10^{-3}m + 1.728 \times 10^{-4}m \]  

(306.27)
Table 306.10: Errors for irregular shaped 8NodeBrick compared to normal shape.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Force direction</th>
<th>Normal shape</th>
<th>Shape 1</th>
<th>Shape 2</th>
<th>Shape 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8NodeBrick</td>
<td>Vertical (z)</td>
<td>0.00%</td>
<td>1.52%</td>
<td>49.33%</td>
<td>33.51%</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>Transverse (y)</td>
<td>0.00%</td>
<td>22.45%</td>
<td>76.20%</td>
<td>18.77%</td>
</tr>
</tbody>
</table>

Figure 306.24: Problem description for cantilever beams under uniform load.

The Real-ESSI displacement results are listed in Table (306.11).

Table 306.11: Results for 8NodeBrick cantilever beams of irregular shapes with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Shape</th>
<th>Force direction</th>
<th>Number of division</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape1</td>
<td>Vertical (z)</td>
<td>5.37E-03 m</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape1</td>
<td>Transverse (y)</td>
<td>4.60E-03 m</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape2</td>
<td>Vertical (z)</td>
<td>2.74E-03 m</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape2</td>
<td>Transverse (y)</td>
<td>1.46E-03 m</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape3</td>
<td>Vertical (z)</td>
<td>9.21E-04 m</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape3</td>
<td>Transverse (y)</td>
<td>1.09E-03 m</td>
</tr>
<tr>
<td>Theoretical solution</td>
<td></td>
<td></td>
<td>7.95E-03 m</td>
</tr>
</tbody>
</table>

The error are listed in Table (306.12).
The errors are shown in Figures (306.25), (306.26) and (306.27).
The Real-ESSI model fei/DSL files for the table above are HERE.
Table 306.12: Errors for 8NodeBrick cantilever beams of irregular shapes with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Shape</th>
<th>Force direction</th>
<th>Number of division</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape1</td>
<td>Vertical (z)</td>
<td>32.42%</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape1</td>
<td>Transverse (y)</td>
<td>42.16%</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape2</td>
<td>Vertical (z)</td>
<td>65.59%</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape2</td>
<td>Transverse (y)</td>
<td>81.57%</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape3</td>
<td>Vertical (z)</td>
<td>88.42%</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>shape3</td>
<td>Transverse (y)</td>
<td>86.24%</td>
</tr>
</tbody>
</table>

Figure 306.25: 8NodeBrick cantilever beam for irregular Shape 1. Displacement error versus Number of division.
Figure 306.26: 8NodeBrick cantilever beam for irregular **Shape 2**. Displacement error versus Number of division.

Figure 306.27: 8NodeBrick cantilever beam for irregular **Shape 3**. Displacement error versus Number of division.
In this section, the beam was cut into smaller elements with element side length 0.5m and 0.25m respectively. And the element side length of the original models is 1.0m. The numerical models are shown in Figures (306.28), (306.29) and (306.30).

Number of division 1:

![Figure 306.28: 8NodeBrick clamped beams with element side length 1.0m.](image)

Number of division 2:

![Figure 306.29: 8NodeBrick clamped beams with element side length 0.5m.](image)

Number of division 4:

![Figure 306.30: 8NodeBrick clamped beams with element side length 0.25m.](image)

The Real-ESSI results are listed in Table (306.13). The theoretical solution is 1.60E-5 m.
The errors are plotted in Figure (306.33).
The Real-ESSI model fei/DSL files for the table above are HERE.
Table 306.13: Results for 8NodeBrick clamped beams with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element side length</th>
<th>1 m</th>
<th>0.5 m</th>
<th>0.25 m</th>
</tr>
</thead>
<tbody>
<tr>
<td>8NodeBrick</td>
<td>1.10E-05 m</td>
<td>1.47E-05 m</td>
<td>1.64E-05 m</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>33.33%</td>
<td>11.09%</td>
<td>0.73%</td>
<td></td>
</tr>
</tbody>
</table>

Figure 306.31: Error scale 0% - 40%.

Figure 306.32: Error scale 0% - 100%.

Figure 306.33: 8NodeBrick clamped beam for different element number. Displacement error versus Number of division.
306.2.7 Verification of 8NodeBrick stress in cantilever beams

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \). Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (306.34).

![Figure 306.34: Problem description for cantilever beams of stress verification.](image)

The theoretical solution for the stress was calculated below.

The 8NodeBrick elements are shown in Figure (306.35).

![Figure 306.35: 8NodeBrick for cantilever beams of stress verification.](image)

The bending moment at the Gaussian Point is

\[
M = F(L - P_y) = 100N \times (6 - 0.2113)m = 578.87N \cdot m
\]

The bending modulus is

\[
I = \frac{bh^3}{12} = \frac{1}{12}m^4
\]

Therefore, the theoretical stress is

\[
\sigma = \frac{M \cdot z}{I} = \frac{578.87N \cdot m \times (0.5 - 0.2113)m}{\frac{1}{12}m^4} = 2005Pa
\]

To get a better result, the same geometry beam was also cut into small elements. When more elements are used, the theoretical stress was calculated again with the new coordinates. The calculation process is similar to the process above.
Figure 306.36: 8NodeBrick stress with element side length 1.0m.

The numerical models are shown in Figures (306.36), (306.37) and (306.38).

Number of division 1:

Number of division 2:

Figure 306.37: 8NodeBrick stress with element side length 0.5m.

Number of division 4:

Figure 306.38: 8NodeBrick stress with element side length 0.25m.

All the stress results are listed in Table (306.14).

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element side length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 m</td>
</tr>
<tr>
<td>8NodeBrick</td>
<td>1270.17 Pa</td>
</tr>
<tr>
<td>Theoretical</td>
<td>2005.26 Pa</td>
</tr>
<tr>
<td>Error</td>
<td>36.66%</td>
</tr>
</tbody>
</table>

The errors are plotted in Figure (306.41).

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.39: Error scale 0% - 40%.

Figure 306.40: Error scale 0% - 100%.

Figure 306.41: 8NodeBrick cantilever beams for stress verification. Stress error versus Number of division.
306.2.8 Verification of 8NodeBrick square plate with four edges clamped

Problem description: Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are clamped.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 N \cdot m \tag{306.33}
\]

The theoretical solution is

\[
d = \alpha_c q a^4 \frac{D}{D} = 0.00406 \times \frac{100 N/m^2 \times 20^4 m^4}{9.1575 \times 10^6 N \cdot m} = 2.2015 \times 10^{-3} m \tag{306.34}
\]

where \( \alpha_c \) is a coefficient, which depends on the ratio of plate length to width. In this problem, the coefficient\(^4\) \( \alpha_c \) is 0.00406.

The 8NodeBrick are shown in Figures (306.42) - (306.47).

\[\text{Figure 306.42: 8NodeBrick edge clamped square plate with element side length 10m.}\]

The results are listed in Table (306.15).

The errors are listed in Table (306.2.8).

The errors are plotted in Figure (306.48).

Figure 306.43: 8NodeBrick edge clamped square plate with element side length 5m.

Table 306.15: Results for 8NodeBrick square plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element side length</th>
<th>Number of layers</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1layer</td>
<td>2layers</td>
</tr>
<tr>
<td>10m</td>
<td>Height:1.00m</td>
<td>Height:0.50m</td>
</tr>
<tr>
<td>9.75E-05 m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5m</td>
<td>3.28E-04 m</td>
<td>3.32E-04 m</td>
</tr>
<tr>
<td>1.04E-03 m</td>
<td>1.10E-03 m</td>
<td>1.12E-03 m</td>
</tr>
<tr>
<td>1m</td>
<td>1.56E-03 m</td>
<td>1.74E-03 m</td>
</tr>
<tr>
<td>1.80E-03 m</td>
<td>2.30E-03 m</td>
<td>2.12E-03 m</td>
</tr>
<tr>
<td>0.5m</td>
<td>1.87E-03 m</td>
<td>2.14E-03 m</td>
</tr>
<tr>
<td>1.87E-03 m</td>
<td>2.14E-03 m</td>
<td>2.23E-03 m</td>
</tr>
</tbody>
</table>

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.44: 8NodeBrick edge clamped square plate with element side length 2m.

<table>
<thead>
<tr>
<th>Element type</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>1layer</td>
<td>2layers</td>
<td>4layers</td>
</tr>
<tr>
<td>Element side length</td>
<td>Height:1.00m</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
</tr>
<tr>
<td>10m</td>
<td>95.57%</td>
<td>95.57%</td>
<td>95.57%</td>
</tr>
<tr>
<td>5m</td>
<td>85.09%</td>
<td>84.94%</td>
<td>84.91%</td>
</tr>
<tr>
<td>2m</td>
<td>52.98%</td>
<td>50.09%</td>
<td>49.25%</td>
</tr>
<tr>
<td>1m</td>
<td>28.93%</td>
<td>21.17%</td>
<td>18.72%</td>
</tr>
<tr>
<td>0.5m</td>
<td>18.26%</td>
<td>4.58%</td>
<td>3.56%</td>
</tr>
<tr>
<td>0.25m</td>
<td>15.05%</td>
<td>2.70%</td>
<td>1.37%</td>
</tr>
</tbody>
</table>

Table 306.16: Errors for 8NodeBrick square plate with four edges clamped.
Figure 306.45: 8NodeBrick edge clamped square plate with element side length 1m.

Figure 306.46: 8NodeBrick edge clamped square plate with element side length 0.5m.
Figure 306.47: 8NodeBrick edge clamped square plate with element side length 0.25m.

Figure 306.48: 8NodeBrick square plate with edge clamped. Displacement error versus Number of side division
306.2.9 Verification of 8NodeBrick square plate with four edges simply supported

Problem description: Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.3$.

The four edges are simply supported.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

$$D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1-0.3^2)} = 9.1575 \times 10^6 N \cdot m$$  \hspace{1cm} (306.35)

The theoretical solution is

$$d = \alpha_s \frac{qa^4}{D} = 0.00126 \times \frac{100N/m^2 \times 20^4 m^4}{9.1575 \times 10^6 N \cdot m} = 7.0936 \times 10^{-3} m$$  \hspace{1cm} (306.36)

where $\alpha_s$ is a coefficient, which depends on the ratio of plate length to width. In this problem, the coefficient$^5 \alpha_s$ is 0.00126.

The 8NodeBrick are shown in Figures (306.49) - (306.54).

---

Figure 306.50: 8NodeBrick edge simply supported square plate with element side length 5m.

Table 306.17: Results for 8NodeBrick square plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element side length</th>
<th>8NodeBrick: Height:0.50m</th>
<th>8NodeBrick: Height:0.25m</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>10m</td>
<td>3.75E-004 m</td>
<td>3.76E-004 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>5m</td>
<td>1.34E-003 m</td>
<td>1.35E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>2m</td>
<td>4.16E-003 m</td>
<td>4.27E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>1m</td>
<td>5.98E-003 m</td>
<td>6.22E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>0.5m</td>
<td>6.75E-003 m</td>
<td>7.04E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>0.25m</td>
<td>8.07E-003 m</td>
<td>7.30E-003 m</td>
<td>7.09E-03 m</td>
</tr>
</tbody>
</table>

The Real-ESSI model fei/DSL files for the table above are [HERE](#).
Figure 306.51: 8NodeBrick edge simply supported square plate with element side length 2m.

Table 306.18: Errors for 8NodeBrick square plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element type</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>2layers</td>
<td>4layers</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
</tr>
<tr>
<td>Element side length</td>
<td>10m</td>
<td>94.72%</td>
<td>94.71%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5m</td>
<td>81.05%</td>
<td>80.91%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2m</td>
<td>41.31%</td>
<td>39.79%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1m</td>
<td>15.64%</td>
<td>12.38%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5m</td>
<td>4.88%</td>
<td>0.70%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.25m</td>
<td>13.74%</td>
<td>2.86%</td>
<td></td>
</tr>
</tbody>
</table>
Figure 306.52: 8NodeBrick edge simply supported square plate with element side length 1m.

Figure 306.53: 8NodeBrick edge simply supported square plate with element side length 0.5m.
Figure 306.54: 8NodeBrick edge simply supported square plate with element side length 0.25m.

Figure 306.55: 8NodeBrick square plate with four edges simply supported. Displacement error versus Number of side division.
306.2.10 Verification of 8NodeBrick circular plate with all edges clamped

Problem description: Diameter=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are clamped.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1-0.3^2)} = 9.1575 \times 10^6 \text{ } N \cdot m \tag{306.37}
\]

The theoretical solution\(^6\) is

\[
d = \frac{qa^4}{64D} = \frac{100N/m^2 \times 10^4 m^4}{64 \times 9.1575 \times 10^6 N \cdot m} = 1.7106 \times 10^{-3} m \tag{306.38}
\]

The 8NodeBrick are shown in Figures (306.56) - (306.61).

Figure 306.57: 8NodeBrick edge clamped circular plate with element side length 5m.

Figure 306.58: 8NodeBrick edge clamped circular plate with element side length 2m.
Figure 306.59: 8NodeBrick edge clamped circular plate with element side length 1m.

Figure 306.60: 8NodeBrick edge clamped circular plate with element side length 0.5m.
Figure 306.61: 8NodeBrick edge clamped circular plate with element side length 0.25m.

Table 306.19: Results for 8NodeBrick circular plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>Number of layers</th>
<th>1layer</th>
<th>2layers</th>
<th>4layers</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of diameter divisions</td>
<td>Height:1.00 m</td>
<td>4</td>
<td>1.97E-04 m</td>
<td>1.99E-04 m</td>
<td>2.00E-04 m</td>
</tr>
<tr>
<td></td>
<td>Height:0.50 m</td>
<td>12</td>
<td>7.95E-04 m</td>
<td>8.47E-04 m</td>
<td>8.62E-04 m</td>
</tr>
<tr>
<td></td>
<td>Height:0.25 m</td>
<td>20</td>
<td>1.13E-03 m</td>
<td>1.25E-03 m</td>
<td>1.28E-03 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>1.36E-03 m</td>
<td>1.54E-03 m</td>
<td>1.60E-03 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60</td>
<td>1.41E-03 m</td>
<td>1.62E-03 m</td>
<td>1.68E-03 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>1.43E-03 m</td>
<td>1.64E-03 m</td>
<td>1.71E-03 m</td>
</tr>
</tbody>
</table>

Figure 306.62: 8NodeBrick circular plate with edge clamped. Displacement error versus Number of side division.
Table 306.20: Errors for 8NodeBrick circular plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>1layer</td>
<td>2layers</td>
<td>4layers</td>
</tr>
<tr>
<td>Number of diameter divisions</td>
<td>Height:1.00\text{m}</td>
<td>Height:0.50\text{m}</td>
<td>Height:0.25\text{m}</td>
</tr>
<tr>
<td>4</td>
<td>88.43%</td>
<td>88.32%</td>
<td>88.30%</td>
</tr>
<tr>
<td>12</td>
<td>53.43%</td>
<td>50.35%</td>
<td>49.47%</td>
</tr>
<tr>
<td>20</td>
<td>33.79%</td>
<td>27.00%</td>
<td>24.93%</td>
</tr>
<tr>
<td>40</td>
<td>20.14%</td>
<td>9.47%</td>
<td>6.03%</td>
</tr>
<tr>
<td>60</td>
<td>17.11%</td>
<td>5.34%</td>
<td>1.51%</td>
</tr>
<tr>
<td>80</td>
<td>16.01%</td>
<td>3.80%</td>
<td>0.19%</td>
</tr>
</tbody>
</table>
306.2.11 Verification of 8NodeBrick circular plate with all edges simply supported

Problem description: Diameter=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are simply supported.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 N \cdot m
\] (306.39)

The theoretical solution\(^7\) is

\[
d = \frac{(5 + \nu)qa^4}{64(1+\nu)D} = \frac{(5 + 0.3) \times 100N/m^2 \times 10^4 m^4}{64 \times (1 + 0.3) \times 9.1575 \times 10^6 N \cdot m} = 6.956 \times 10^{-3} m
\] (306.40)

The 8NodeBrick are shown in Figures \((306.63) - (306.68)\).

![8NodeBrick edge simply supported circular plate with element side length 10m.](image)

Figure 306.63: 8NodeBrick edge simply supported circular plate with element side length 10m.

The results are listed in Table \((306.21)\).

The errors are listed in Table \((306.22)\).

The errors are plotted in Figure \((306.69)\).

The Real-ESSI model fei/DSL files for the table above are HERE.

Figure 306.64: 8NodeBrick edge simply supported circular plate with element side length 5m.

Figure 306.65: 8NodeBrick edge simply supported circular plate with element side length 2m.
Figure 306.66: 8NodeBrick edge simply supported circular plate with element side length 1m.

Figure 306.67: 8NodeBrick edge simply supported circular plate with element side length 0.5m.
Figure 306.68: 8NodeBrick edge simply supported circular plate with element side length 0.25m.

Table 306.21: Results for 8NodeBrick circular plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element type</th>
<th>8NodeBrick (2 layers)</th>
<th>8NodeBrick (4 layers)</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>Height:0.50 m</td>
<td>Height:0.25 m</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6.35E-04 m</td>
<td>6.39E-04 m</td>
<td>6.96E-03 m</td>
</tr>
<tr>
<td>12</td>
<td>3.46E-03 m</td>
<td>3.57E-03 m</td>
<td>6.96E-03 m</td>
</tr>
<tr>
<td>20</td>
<td>4.96E-03 m</td>
<td>5.18E-03 m</td>
<td>6.96E-03 m</td>
</tr>
<tr>
<td>40</td>
<td>6.05E-03 m</td>
<td>6.37E-03 m</td>
<td>6.96E-03 m</td>
</tr>
<tr>
<td>60</td>
<td>6.30E-03 m</td>
<td>6.65E-03 m</td>
<td>6.96E-03 m</td>
</tr>
<tr>
<td>80</td>
<td>6.39E-03 m</td>
<td>6.76E-03 m</td>
<td>6.96E-03 m</td>
</tr>
</tbody>
</table>
Table 306.22: Errors for 8NodeBrick circular plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element type</th>
<th>8NodeBrick</th>
<th>8NodeBrick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>2layers</td>
<td>4layers</td>
</tr>
<tr>
<td>Number of diameter divisions</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
</tr>
<tr>
<td>4</td>
<td>90.87%</td>
<td>90.82%</td>
</tr>
<tr>
<td>12</td>
<td>50.19%</td>
<td>48.65%</td>
</tr>
<tr>
<td>20</td>
<td>28.64%</td>
<td>25.47%</td>
</tr>
<tr>
<td>40</td>
<td>13.09%</td>
<td>8.40%</td>
</tr>
<tr>
<td>60</td>
<td>9.45%</td>
<td>4.36%</td>
</tr>
<tr>
<td>80</td>
<td>8.10%</td>
<td>2.85%</td>
</tr>
</tbody>
</table>

Figure 306.69: 8NodeBrick circular plate with edge simply supported. Displacement error versus Number of side division.
306.2.12 Verification of 8NodeBrick Finite Element for Boussinesq Problem

306.2.12.1 Introduction

The Boussinesq problem is finding the displacement distribution in the isotropic linearly elastic half-space, subject to a concentrated load applied on the surface and perpendicular to it. The Boussinesq problem diagram is shown in Fig. (306.166).

Boussinesq problem is widely used in geotechnical engineering, especially when designing a foundation which transfers the superstructure load to the soil. To estimate the foundation settlements, it is important to have a reliable numerical solution for the Boussinesq problem.

![Boussinesq problem diagram](image)

**Figure 306.70:** Boussinesq problem description. (Figure Reference: Verruijt, Arnold, and Stefan Van Baars. Soil mechanics. Delft, 2007.)

In 1885, the French scientist Joseph Boussinesq solved the analytic solutions of displacements in the homogeneous isotropic linear elastic half space. In general, the vertical displacement of the surface is

\[
z = 0 : \quad u_z = \frac{P(1 - \nu^2)}{\pi ER}
\]

(306.41)

where \( P \) is the vertical load, \( \nu \) is the Poisson’s ratio, \( E \) is the elastic modulus, and \( R \) is the distance from the measured point to the loading point.

In this section, the Real-ESSI numerical solution is verified by the analytic solution for the Boussinesq problem.

306.2.12.2 Description of the Verification Model

Since the problem is cylindrical symmetry, a quarter of the entire cube was employed to represent the whole cube. The reduced model was shown in Fig. (306.167).
The side length is 20 meters and the load $P$ is 1N. The elastic modulus $E = 1 \times 10^3 Pa$ and the Poisson’s ratio $\nu = 0.0$.

![Figure 306.71: Reduced model (One quarter model) for the point load on the half space](image)

The boundary conditions are shown in Fig. (306.72) and (306.73).

![Figure 306.72: X-Z view for the reduced model](image)

### 306.2.12.3 Results

#### Analytic solution for this model

According to the previous introduction, the analytic solution on the surface for this problem is

$$z = 0: \quad u_z = \frac{P(1 - \nu^2)}{\pi E R} = \frac{1}{10^3 \pi} \frac{1}{R}$$

(306.42)

On the face $x = 0$, the distance $R$ on the surface is actually the value of $y$, therefore, the analytic
As long as the $y$ values are substituted, the displacement $u_z$ is obtained immediately.

**Real-ESSI solution with 8NodeBrickLT**

In Real-ESSI, 8NodeBrickLT elements were used to simulate this model. Each element is $2m \times 2m \times 2m$. Since the model is $20m \times 20m \times 20m$, the element number is $10 \times 10 \times 10 = 1000$. The vertical displacement at the surface was recorded.

Since the model is symmetric, when the results were plotted, the other half results were obtained by symmetry.

**Comparison between the analytic and 8NodeBrickLT solution**

The Real-ESSI and analytic results were plotted in Fig. (306.74). Note that the analytic solution for location $y = 0$ is infinity, which was not plotted in the figure below.

### 306.2.12.4 Error Analysis

1. **Mismatch at the loading point.**

   First of all, when $x = 0$ at the loading point, the analytic solution is infinite. From the perspective of practical engineering, this analytic solution is flawed because the displacement cannot be infinite. The infinite solution is due to the elastic assumption. In consideration of the plasticity, the analytic solution will not be infinite.

   In Real-ESSI, the displacement at the loading point is not infinite because the infinite value is averaged by the integration during the finite element calculation. Also, at the loading point,
27NodeBrickLT has a much larger displacement than that of 8NodeBrickLT. This is because 27NodeBrickLT has a relatively denser mesh than 8NodeBrickLT. So the maximum value at the loading point is higher than that of 8NodeBrickLT.

2. Mismatch at other locations.

Except at the loading point, the analytic solution is not exactly equal to the numerical solution at other locations. This is because the verification example employs a simplified bounded cube to represent unbounded half space. The original analytic solution is for the half space, which is not true for the verification model. Not only the horizontal space but also the bottom space are removed from the model. This means the analytic solution is not perfect for this bounded cube. However, since the cube is very great, the analytic solution is similar to the Real-ESSI numerical solution. In addition, the brick elements are also verified by other models, like beam, plate and shells.
306.2.13 Verification of 8NodeBrick Finite Element for Collapsed Brick Shapes

306.2.13.1 Test procedure

With reference to the “patch test” put forward by Taylor et al. (1986), the test procedures are:

- A standard solution is given by testing two different loading modes on a single normal 8 node brick element: (1) Pure confinement loading, where same pressure are applied on three different directions. (2) Simple shearing, where shearing force is applied on four nodes of top layer, while four nodes on bottom layer are totally fixed. Linear elastic material is adopted here with Young’s modulus $E = 125\text{MPa}$ and Poisson’s ratio $\mu = 0.25$. The setup of standard test is shown in Figure 306.75.

- Build the same geometric model with collapsed 8 node brick elements and conduct numerical simulation under the same loading and boundary conditions as first step.

Specifically, the geometric configure for 7-node collapsed element is shown in Figure 306.76, where the cubic consists of two 7-node collapsed elements. A dummy node 11 is generated at the same location as node 2.

The geometric configuration for 6-node collapsed element is shown in Figure 306.77. Again the cubic is composed of two 6-node collapsed elements. Two dummy Real-ESSI nodes 9 and 10 are
Figure 306.76: Geometric configuration of numerical test for 7-node collapsed element.

generated at the same location with original node 1 and 2.

And Figure 306.78 gives the geometric configuration of 5-node collapsed element, where the cubic is divided into 3 5-node elements.

- Compare the response of models in step 1 and step 2. If the difference is small enough, the strategy described in section ?? is feasible and valid.

306.2.13.2 Test result

The comparison of displacement response for confinement loading is shown in Figure 306.79. Figure 306.80 demonstrates the test results of simple shearing loading. It can be seen that the simulation results of these types of collapsed element are close to result of standard 8 node brick element. The line of 6 node collapsed element is almost overlap with the line of standard test. Collapsed 7-node element and 5-node element experience certain decrease of accuracy. The main error reflects on the decrease of stiffness. Both bulk modulus and shear modulus of 7-node element and 5-node element are around 7% lower than the standard 8-node element.
Figure 306.77: Geometric configuration of numerical test for 6-node collapsed element.

Figure 306.78: Geometric configuration of numerical test for 5-node collapsed element.
Figure 306.79: Comparison of displacement response under confinement loading.

Figure 306.80: Comparison of displacement response under simple shearing.
306.2.14 Verification of 27 node brick cantilever beam (static)

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \). Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (306.81).

![Figure 306.81: Problem description for cantilever beams.](image)

Theoretical displacement (bending and shear deformation):

\[
d = \frac{FL^3}{3EI} + \frac{FL}{GA_v} = \frac{FL^3}{3E\frac{bh^3}{12}} + \frac{FL}{E\frac{bh}{2(1+\nu)}} = \frac{100N \times 6^3m^3}{3 \times 10^8N/m^2 \times \frac{1m^4}{12m^4}} + \frac{100N \times 6m}{\frac{10^7N/m^2 \times 1m^2 \times \frac{5}{6}}{m^2}} = 8.64 \times 10^{-4}m + 0.144 \times 10^{-4}m \tag{306.44}
\]

Numerical model:

The 27NodeBrick elements are shown in Figure (306.85).

All the Real-ESSI results are listed in Table (306.23).

<table>
<thead>
<tr>
<th>Element number</th>
<th>1</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>27NodeBrick</td>
<td>7.07E-04 m</td>
<td>8.50E-04 m</td>
<td>8.75E-04 m</td>
</tr>
<tr>
<td>Error</td>
<td>19.52%</td>
<td>3.19%</td>
<td>0.34%</td>
</tr>
</tbody>
</table>

The errors are plotted in Figure (306.88).

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.82: One 27NodeBrick element.

Figure 306.83: Two 27NodeBrick elements.

Figure 306.84: Six 27NodeBrick elements.

Figure 306.85: 27NodeBrick elements for cantilever beams.
Figure 306.86: Error scale 0% - 20%.

Figure 306.87: Error scale 0% - 100%.

Figure 306.88: 27NodeBrick cantilever beam for different element number. Displacement error versus Number of elements.
306.2.15 Verification of 27NodeBrick cantilever beam for different Poisson’s ratio

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.0 - 0.49$.
The force direction was shown in Figure (306.89).

![Figure 306.89: Problem description for cantilever beams of different Poisson’s ratios.](image)

The theoretical solution for $\nu = 0.0$ was calculated below, while the solution for other Poisson’s ratio
is calculated by the similar process.

Theoretical displacement (bending and shear deformation):

$$d = \frac{FL^3}{3EI} + \frac{FL}{GA_v}$$

$$= \frac{FL^3}{3EI} + \frac{FL}{2(1+\nu) \frac{bh}{12}}$$

$$= \frac{100N \times 6^3 m^3}{3 \times 10^8 N/m^2 \times \frac{1}{12} m^4} + \frac{100N \times 6m}{\frac{10}{2} \times 10^7 N/m^2 \times 1m^2 \times \frac{5}{6}}$$

$$= 8.64 \times 10^{-4} m + 0.144 \times 10^{-4} m$$

$$= 8.784 \times 10^{-4} m$$

(306.45)

The rotation angle at the end:

$$\theta = \frac{FL^2}{2EI} = \frac{100N \times 6^2 m^2}{2 \times 10^8 N/m^2 \times \frac{1}{12} m^4} = 2.16 \times 10^{-4} \text{ rad} = 0.0124^\circ$$

(306.46)

The 27NodeBrick elements for cantilever beams of different Poisson’s ratios are shown in Figure (306.90).

All the displacement results are listed in Table (306.24).
The errors are plotted in Figure (306.93).
The angle results are listed in Table (306.27).
The errors are plotted in Figure (306.96).
Figure 306.90: 27NodeBrick elements for cantilever beams of different Poisson’s ratios.

Table 306.24: **Displacement** results for 27NodeBrick cantilever beams with element side length 1 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>27NodeBrick displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.755E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>0.34%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.757E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>0.39%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.751E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>0.54%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.735E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>0.80%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.708E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>1.19%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.667E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>1.74%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.608E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>2.50%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.520E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>3.57%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.385E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>5.18%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.147E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>7.96%</td>
</tr>
<tr>
<td>0.49</td>
<td>7.711E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>12.94%</td>
</tr>
</tbody>
</table>

The Real-ESSI model fei/DSL files for the table above are [HERE](#).

Then, different values of elastic modulus were also tried. The errors are plotted below. According to Fig. (306.97)), the different values of elastic modulus will not influence the error.

However, the different Poisson’s ratio will influence the error. The error will increase with the Poisson’s ratio increase.
Table 306.25: **Displacement** results for 27NodeBrick cantilever beams with element side length 0.5 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>27NodeBrick displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement (all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.804E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>0.23%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.808E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>0.19%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.805E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>0.08%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.796E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>0.12%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.778E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>0.40%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.752E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>0.78%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.715E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>1.28%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.663E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>1.95%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.588E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>2.89%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.465E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>4.36%</td>
</tr>
<tr>
<td>0.49</td>
<td>8.248E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>6.88%</td>
</tr>
</tbody>
</table>

Table 306.26: **Displacement** results for 27NodeBrick cantilever beams with element side length 0.25 m.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>27NodeBrick displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement (all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.797E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>0.15%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.801E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>0.11%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.799E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>0.01%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.792E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>0.16%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.778E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>0.40%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.758E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>0.71%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.730E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>1.12%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.692E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>1.63%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.641E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>2.29%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.567E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>3.21%</td>
</tr>
<tr>
<td>0.49</td>
<td>8.452E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>4.58%</td>
</tr>
</tbody>
</table>
Figure 306.91: Error scale 0% - 15%.

Figure 306.92: Error scale 0% - 100%.

Figure 306.93: 27NodeBrick cantilever beam for different Poisson’s ratio. Displacement error versus Poisson’s ratio
Table 306.27: Rotation angle results for 27NodeBrick cantilever beams with element side length 1 m.

<table>
<thead>
<tr>
<th>Poisson's ratio</th>
<th>27NodeBrick angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.238E-02</td>
<td>1.24E-02</td>
<td>0.19%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.237E-02</td>
<td>1.24E-02</td>
<td>0.24%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.236E-02</td>
<td>1.24E-02</td>
<td>0.34%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.233E-02</td>
<td>1.24E-02</td>
<td>0.53%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.230E-02</td>
<td>1.24E-02</td>
<td>0.80%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.225E-02</td>
<td>1.24E-02</td>
<td>1.18%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.219E-02</td>
<td>1.24E-02</td>
<td>1.70%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.210E-02</td>
<td>1.24E-02</td>
<td>2.45%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.196E-02</td>
<td>1.24E-02</td>
<td>3.55%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.172E-02</td>
<td>1.24E-02</td>
<td>5.47%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.130E-02</td>
<td>1.24E-02</td>
<td>8.89%</td>
</tr>
</tbody>
</table>

Table 306.28: Rotation angle results for 27NodeBrick cantilever beams with element side length 0.5 m.

<table>
<thead>
<tr>
<th>Poisson's ratio</th>
<th>27NodeBrick angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.242E-02</td>
<td>1.24E-02</td>
<td>0.12%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.241E-02</td>
<td>1.24E-02</td>
<td>0.11%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.241E-02</td>
<td>1.24E-02</td>
<td>0.06%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.239E-02</td>
<td>1.24E-02</td>
<td>0.05%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.237E-02</td>
<td>1.24E-02</td>
<td>0.21%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.235E-02</td>
<td>1.24E-02</td>
<td>0.44%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.231E-02</td>
<td>1.24E-02</td>
<td>0.74%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.226E-02</td>
<td>1.24E-02</td>
<td>1.16%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.218E-02</td>
<td>1.24E-02</td>
<td>1.76%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.206E-02</td>
<td>1.24E-02</td>
<td>2.76%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.183E-02</td>
<td>1.24E-02</td>
<td>4.63%</td>
</tr>
</tbody>
</table>
Table 306.29: **Rotation angle** results for 27NodeBrick cantilever beams with element side length 0.25 m.

<table>
<thead>
<tr>
<th>Poisson's ratio</th>
<th>27NodeBrick angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.242E-02</td>
<td>1.24E-02</td>
<td>0.17%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.242E-02</td>
<td>1.24E-02</td>
<td>0.15%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.241E-02</td>
<td>1.24E-02</td>
<td>0.09%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.240E-02</td>
<td>1.24E-02</td>
<td>0.02%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.238E-02</td>
<td>1.24E-02</td>
<td>0.17%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.235E-02</td>
<td>1.24E-02</td>
<td>0.38%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.232E-02</td>
<td>1.24E-02</td>
<td>0.64%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.228E-02</td>
<td>1.24E-02</td>
<td>0.98%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.222E-02</td>
<td>1.24E-02</td>
<td>1.42%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.214E-02</td>
<td>1.24E-02</td>
<td>2.06%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.202E-02</td>
<td>1.24E-02</td>
<td>3.08%</td>
</tr>
</tbody>
</table>
Figure 306.94: Error scale 0% - 10%.

Figure 306.95: Error scale 0% - 100%.

Figure 306.96: 27NodeBrick cantilever beam for different Poisson’s ratio. Rotation angle error versus Poisson’s ratio

Figure 306.97: The influence of Poisson’s ratio and elastic modulus on the errors.
306.2.16 Test of irregular shaped 27NodeBrick cantilever beams

Cantilever model was used as an example. Three different shapes are tested.

In the first test, the upper two nodes of each element were moved one half element size along the $y$-axis, while the lower two nodes were kept at the same location. The element shape was shown in Figure (306.98).

![Figure 306.98: 27NodeBrick cantilever beams for irregular Shape 1.](image)

In the second test, the upper two nodes of each element were moved 90% element size along the $y$-axis, while the lower two nodes were moved 90% element size in the other direction along the $y$-axis. The element shape was shown in Figure (306.99).

![Figure 306.99: 27NodeBrick cantilever beams for irregular Shape 2.](image)

In the third test, the upper two nodes of each element are moved one half element size with different directions along the $y$-axis, while the lower two nodes were kept at the same location. The element shape was shown in Figure (306.100).

![Figure 306.100: 27NodeBrick cantilever beams for irregular Shape 3.](image)

The boundary conditions are shown in Figure (306.103), (306.106) and (306.109).

The Real-ESSI results are listed in Table (306.30).

The errors are listed in Table (306.31) and (306.32).
Figure 306.101: Vertical force.

Figure 306.102: Horizontal force.

Figure 306.103: 27NodeBrick cantilever beam boundary conditions for irregular Shape 1.

Figure 306.104: Vertical force.

Figure 306.105: Horizontal force.

Figure 306.106: 27NodeBrick cantilever beam boundary conditions for irregular Shape 2.

Table 306.30: Results for 27NodeBrick cantilever beams of irregular shapes.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Force direction</th>
<th>Normal shape</th>
<th>Shape 1</th>
<th>Shape 2</th>
<th>Shape 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>27NodeBrick</td>
<td>Vertical (z)</td>
<td>8.755E-04 m</td>
<td>8.819E-04 m</td>
<td>8.709E-04 m</td>
<td>8.837E-04 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>Transverse (y)</td>
<td>8.755E-04 m</td>
<td>8.831E-04 m</td>
<td>8.462E-04 m</td>
<td>8.824E-04 m</td>
</tr>
<tr>
<td>Theoretical</td>
<td>-</td>
<td>8.784E-04 m</td>
<td>8.784E-04 m</td>
<td>8.784E-04 m</td>
<td>8.784E-04 m</td>
</tr>
</tbody>
</table>
Figure 306.107: Vertical force.

Figure 306.108: Horizontal force.

Figure 306.109: 27NodeBrick cantilever beam boundary conditions for irregular Shape 3.

Table 306.31: Errors for irregular shaped 27NodeBrick compared to theoretical solution.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Force direction</th>
<th>Normal shape</th>
<th>Shape 1</th>
<th>Shape 2</th>
<th>Shape 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>27NodeBrick</td>
<td>Vertical (z)</td>
<td>0.34%</td>
<td>0.40%</td>
<td>0.85%</td>
<td>0.60%</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>Transverse (y)</td>
<td>0.34%</td>
<td>0.54%</td>
<td>3.67%</td>
<td>0.46%</td>
</tr>
</tbody>
</table>

Table 306.32: Errors for irregular shaped 27NodeBrick compared to normal shape.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Force direction</th>
<th>Normal shape</th>
<th>Shape 1</th>
<th>Shape 2</th>
<th>Shape 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>27NodeBrick</td>
<td>Vertical (z)</td>
<td>0.00%</td>
<td>0.74%</td>
<td>0.52%</td>
<td>0.94%</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>Transverse (y)</td>
<td>0.00%</td>
<td>0.87%</td>
<td>3.34%</td>
<td>0.79%</td>
</tr>
</tbody>
</table>

The Real-ESSI model fei/DSL files for the table above are HERE.

Then, the beam was divided into small elements.

Problem description: Length=12m, Width=2m, Height=2m, Force=400N/m, \( E=1E8 \)Pa, \( \nu = 0.0 \).

Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (306.110).
Figure 306.110: Problem description for cantilever beams under uniform pressure.

Theoretical displacement (bending and shear deformation):

\[
d = \frac{qL^4}{8EI} + \frac{qL^2}{GA_v}
\]

\[
= \frac{qL^4}{8Eh^3/T^2} + \frac{qL^2}{2(1+\nu)bh} \frac{bh}{T^2}
\]

\[
= \frac{400N/m \times 12^4m^4}{8 \times 10^8N/m^2 \times 2^4m^4} + \frac{10^8N/m^2 \times 2m \times 2m \times \frac{5}{6}}{2^2N/m^2 \times 2m \times 2m \times \frac{5}{6}}
\]

\[
= 7.776 \times 10^{-3}m + 1.728 \times 10^{-4}m
\]

\[
= 7.9488 \times 10^{-3}m
\]

The Real-ESSI displacement results are listed in Table (306.33).

Table 306.33: Results for 27NodeBrick cantilever beams of irregular shapes with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Shape</th>
<th>Force direction</th>
<th>Number of division</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>27NodeBrick</td>
<td>shape1</td>
<td>Vertical (z)</td>
<td></td>
<td>7.913E-03 m</td>
<td>7.946E-03 m</td>
<td>7.948E-03 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>shape1</td>
<td>Transverse (y)</td>
<td></td>
<td>7.903E-03 m</td>
<td>7.946E-03 m</td>
<td>7.948E-03 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>shape2</td>
<td>Vertical (z)</td>
<td></td>
<td>7.741E-03 m</td>
<td>7.930E-03 m</td>
<td>7.947E-03 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>shape2</td>
<td>Transverse (y)</td>
<td></td>
<td>7.371E-03 m</td>
<td>7.894E-03 m</td>
<td>7.944E-03 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>shape3</td>
<td>Vertical (z)</td>
<td></td>
<td>1.982E-03 m</td>
<td>7.946E-03 m</td>
<td>7.948E-03 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>shape3</td>
<td>Transverse (y)</td>
<td></td>
<td>1.979E-03 m</td>
<td>7.947E-03 m</td>
<td>7.948E-03 m</td>
</tr>
<tr>
<td>Theoretical solution</td>
<td></td>
<td></td>
<td></td>
<td>7.9488E-03 m</td>
<td>7.9488E-03 m</td>
<td>7.9488E-03 m</td>
</tr>
</tbody>
</table>

The error are listed in Table (306.34).

The errors are shown in Figures (306.113), (306.116) and (306.119).

The Real-ESSI model fei/DSL files for the table above are HERE.
Table 306.34: Errors for 27NodeBrick cantilever beams of irregular shapes with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Shape</th>
<th>Force direction</th>
<th>Number of division</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 306.111: Error scale 0% - 0.4%.

Figure 306.112: Error scale 0% - 100%.

Figure 306.113: 27NodeBrick cantilever beam for irregular **Shape 1**. Displacement error versus Number of division.

\( \nu = 0.0 \). Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (??).

In this section, the beam was cut into smaller elements with element side length 0.5m and 0.25m respectively. And the element side length of the original models is 1.0m. The numerical models are shown in Figure (306.120), (306.121) and (306.122).
The Real-ESSI results are listed in Table (306.35). The theoretical solution is $1.60 \times 10^{-5} m$.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element side length</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 m</td>
<td>0.5 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>1.64E-05 m</td>
<td>1.70E-05 m</td>
</tr>
<tr>
<td>Error</td>
<td>0.83%</td>
<td>3.25%</td>
</tr>
</tbody>
</table>

The errors are plotted in Figure (306.125).

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.117: Error scale 0% - 80%.

Figure 306.118: Error scale 0% - 100%.

Figure 306.119: 27NodeBrick cantilever beam for irregular Shape 3. Displacement error versus Number of division

Figure 306.120: 27NodeBrick clamped beams with element side length 1.0m.

Figure 306.121: 27NodeBrick clamped beams with element side length 0.5m.
Figure 306.122: 27NodeBrick clamped beams with element side length 0.25m.

Figure 306.123: Error scale 0% - 4%.

Figure 306.124: Error scale 0% - 100%.

Figure 306.125: 27NodeBrick clamped beam for different element number. Displacement error versus Number of division.
306.2.17 Verification of 27NodeBrick stress in cantilever beams

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \). Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (306.126).

![Figure 306.126: Problem description for cantilever beams of stress verification.](image)

The theoretical solution for the stress was calculated below.

The 27NodeBrick elements are shown in Figure (306.127).

![Figure 306.127: 27NodeBrick for cantilever beams of stress verification.](image)

The bending moment at the Gaussian Point is

\[
M = F(L - P_y) = 100N \times (6 - 0.1127)m = 588.73N \cdot m
\]

(306.48)

The bending modulus is

\[
I = \frac{bh^3}{12} = \frac{1}{12}m^4
\]

(306.49)

Therefore, the theoretical stress is

\[
\sigma = \frac{M \cdot z}{I} = \frac{588.73N \cdot m \times (0.5 - 0.1127)m}{\frac{1}{12}m^4} = 2736Pa
\]

(306.50)

To get a better result, the same geometry beam was also cut into small elements. When more elements are used, the theoretical stress was calculated again with the new coordinates. The calculation process is similar to the process above.
The numerical models are shown in Figure (306.128), (306.129) and (306.130).

Number of division 1:

Number of division 2:

All the stress results are listed in Table (306.36).

Table 306.36: Results for 27NodeBrick stress with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element side length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>2719.81 Pa</td>
</tr>
<tr>
<td>Theoretical</td>
<td>2736.17 Pa</td>
</tr>
<tr>
<td>Error</td>
<td>0.60%</td>
</tr>
</tbody>
</table>

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.131: Error scale 0% - 2.5%.

Figure 306.132: Error scale 0% - 100%.

Figure 306.133: 27NodeBrick cantilever beams for stress verification. Stress error versus Number of division.
306.2.18 Verification of 27NodeBrick square plate with four edges clamped

Problem description: Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are clamped.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3m^3}{12 \times (1-0.3^2)} = 9.1575 \times 10^6 N \cdot m
\]  

(306.51)

The theoretical solution is

\[
d = \alpha_c \frac{qa^4}{D} = 0.00406 \times \frac{100N/m^2 \times 20^4m^4}{9.1575 \times 10^6 N \cdot m} = 2.2015 \times 10^{-3}m
\]

(306.52)

where \( \alpha_c \) is a coefficient, which depends on the ratio of plate length to width. In this problem, the coefficient\(^8 \) \( \alpha_c \) is 0.00406.

The 27NodeBrick are shown in Figure (306.134) - (306.139).

![27NodeBrick edge clamped square plate with element side length 10m.](image)

Figure 306.134: 27NodeBrick edge clamped square plate with element side length 10m.

The results were listed in Table (306.37).

---


\(^9\) This model run out of memory on machine cml01 (memory: 23.5GB). This model has 233,289 nodes with 3 dofs, which may require 40GB memory.
Figure 306.135: 27NodeBrick edge clamped square plate with element side length 5m.

Table 306.37: Results for 27NodeBrick square plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>1layer</td>
<td>2layers</td>
<td>4layers</td>
<td></td>
</tr>
<tr>
<td>Element side length</td>
<td>Height:1.00m</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
<td></td>
</tr>
<tr>
<td>10m</td>
<td>4.82E-004 m</td>
<td>4.82E-004 m</td>
<td>4.82E-004 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>5m</td>
<td>1.97E-003 m</td>
<td>1.98E-003 m</td>
<td>1.98E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>2m</td>
<td>2.25E-003 m</td>
<td>2.26E-003 m</td>
<td>2.26E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>1m</td>
<td>2.28E-003 m</td>
<td>2.29E-003 m</td>
<td>2.29E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>0.5m</td>
<td>2.29E-003 m</td>
<td>2.30E-003 m</td>
<td>2.30E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>0.25m</td>
<td>2.29E-003 m</td>
<td>2.30E-003 m</td>
<td>-9</td>
<td>2.20E-03 m</td>
</tr>
</tbody>
</table>

The errors were listed in Table (306.38).
The errors were plotted in Figure (306.140).
The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.136: 27NodeBrick edge clamped square plate with element side length 2m.

Table 306.38: Errors for 27NodeBrick square plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>1layer</td>
<td>2layers</td>
<td>4layers</td>
</tr>
<tr>
<td>Element side length</td>
<td>Height:1.00m</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
</tr>
<tr>
<td>10m</td>
<td>78.11%</td>
<td>78.10%</td>
<td>78.10%</td>
</tr>
<tr>
<td>5m</td>
<td>10.67%</td>
<td>10.19%</td>
<td>10.16%</td>
</tr>
<tr>
<td>2m</td>
<td>2.23%</td>
<td>2.79%</td>
<td>2.83%</td>
</tr>
<tr>
<td>1m</td>
<td>3.56%</td>
<td>4.16%</td>
<td>4.22%</td>
</tr>
<tr>
<td>0.5m</td>
<td>3.96%</td>
<td>4.58%</td>
<td>4.65%</td>
</tr>
<tr>
<td>0.25m</td>
<td>4.08%</td>
<td>4.70%</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 306.137: 27NodeBrick edge clamped square plate with element side length 1m.

Figure 306.138: 27NodeBrick edge clamped square plate with element side length 0.5m.
Figure 306.139: 27Node Brick edge clamped square plate with element side length 0.25m.

Figure 306.140: 27Node Brick square plate with edge clamped. Displacement error versus Number of side division.
306.2.19 Verification of 27NodeBrick square plate with four edges simply supported

Problem description: Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.3$.

The four edges are simply supported.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

$$D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 N \cdot m$$ (306.53)

The theoretical solution is

$$d = \alpha_s \frac{qa^4}{D} = 0.00126 \times \frac{100 N/m^2 \times 20^4 m^4}{9.1575 \times 10^6 N \cdot m} = 7.0936 \times 10^{-3} m$$ (306.54)

where $\alpha_s$ is a coefficient, which depends on the ratio of plate length to width. In this problem, the coefficient$^{10} \alpha_s$ is 0.00126.

The 27NodeBrick were shown in Figure (306.141) - (306.146).

![Figure 306.141: 27NodeBrick edge simply supported square plate with element side length 10m.](image)

The results were listed in Table (306.39).

---


$^{11}$This model run out of memory on machine cml01 (memory: 23.5GB). This model has 233,289 nodes with 3 dofs, which may require 40GB memory.
Figure 306.142: 27NodeBrick edge simply supported square plate with element side length 5m.

Table 306.39: Results for 27NodeBrick square plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element type</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>2layers</td>
<td>4layers</td>
<td></td>
</tr>
<tr>
<td>Element side length</td>
<td>Height: 0.50m</td>
<td>Height: 0.25m</td>
<td></td>
</tr>
<tr>
<td>10m</td>
<td>6.54E-003 m</td>
<td>6.54E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>5m</td>
<td>7.24E-003 m</td>
<td>7.24E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>2m</td>
<td>7.44E-003 m</td>
<td>7.44E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>1m</td>
<td>7.49E-003 m</td>
<td>7.49E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>0.5m</td>
<td>7.50E-003 m</td>
<td>7.50E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>0.25m</td>
<td>7.51E-003 m</td>
<td>-</td>
<td>7.09E-03 m</td>
</tr>
</tbody>
</table>

The errors were listed in Table (306.40).
The errors were plotted in Figure (306.149).
The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 306.143: 27NodeBrick edge simply supported square plate with element side length 2m.

Table 306.40: Errors for 27NodeBrick square plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element side length</th>
<th>Height:0.50m</th>
<th>Height:0.25m</th>
</tr>
</thead>
<tbody>
<tr>
<td>10m</td>
<td>7.87%</td>
<td>7.85%</td>
</tr>
<tr>
<td>5m</td>
<td>2.07%</td>
<td>2.10%</td>
</tr>
<tr>
<td>2m</td>
<td>4.85%</td>
<td>4.89%</td>
</tr>
<tr>
<td>1m</td>
<td>5.54%</td>
<td>5.58%</td>
</tr>
<tr>
<td>0.5m</td>
<td>5.74%</td>
<td>5.79%</td>
</tr>
<tr>
<td>0.25m</td>
<td>5.80%</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 306.144: 27NodeBrick edge simply supported square plate with element side length 1m.

Figure 306.145: 27NodeBrick edge simply supported square plate with element side length 0.5m.
Figure 306.146: 27NodeBrick edge simply supported square plate with element side length 0.25m.
Figure 306.147: Error scale 0% - 8%.

Figure 306.148: Error scale 0% - 100%.

Figure 306.149: 27NodeBrick square plate with edge simply supported. Displacement error versus Number of side division.
306.2.20 Verification of 27NodeBrick circular plate with all edges clamped

Problem description: Diameter=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are clamped.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 N \cdot m
\]  

(306.55)

The theoretical solution\(^{12}\) is

\[
d = \frac{qa^4}{64D} = \frac{100 N/m^2 \times 10^4 m^4}{64 \times 9.1575 \times 10^6 N \cdot m} = 1.7106 \times 10^{-3} m
\]  

(306.56)

The 27NodeBrick were shown in Figure (306.150) - (306.155).

![Figure 306.150: 27NodeBrick edge clamped circular plate with element side length 10m.](image)

The results were listed in Table (306.41).

The errors were listed in Table (306.42).

The errors were shown in Figure (306.156).

The Real-ESSI model fei/DSL files for the table above are HERE.

---

Figure 306.151: 27NodeBrick edge clamped circular plate with element side length 5m.

Figure 306.152: 27NodeBrick edge clamped circular plate with element side length 2m.
Figure 306.153: 27NodeBrick edge clamped circular plate with element side length 1m.

Figure 306.154: 27NodeBrick edge clamped circular plate with element side length 0.5m.
Figure 306.155: 27NodeBrick edge clamped circular plate with element side length 0.25m.

Table 306.41: Results for 27NodeBrick circular plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>Number of layers</th>
<th>Number of diameter divisions</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>27NodeBrick</td>
<td>1layer</td>
<td>Height:1.00m</td>
<td>2.777E-03 m</td>
</tr>
<tr>
<td></td>
<td>2layers</td>
<td>Height:0.50m</td>
<td>2.786E-03 m</td>
</tr>
<tr>
<td></td>
<td>4layers</td>
<td>Height:0.25m</td>
<td>2.789E-03 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.706E-03 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>2layers</td>
<td>Height:0.50m</td>
<td>2.786E-03 m</td>
</tr>
<tr>
<td></td>
<td>4layers</td>
<td>Height:0.25m</td>
<td>2.789E-03 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.706E-03 m</td>
</tr>
<tr>
<td>27NodeBrick</td>
<td>4layers</td>
<td>Height:0.25m</td>
<td>2.789E-03 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.706E-03 m</td>
</tr>
</tbody>
</table>
Table 306.42: Errors for 27NodeBrick circular plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>1layer</td>
<td>2layers</td>
<td>4layers</td>
</tr>
<tr>
<td>Number of diameter divisions</td>
<td>Height:1.00m</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
</tr>
<tr>
<td>4</td>
<td>62.75%</td>
<td>63.42%</td>
<td>63.47%</td>
</tr>
<tr>
<td>12</td>
<td>62.46%</td>
<td>63.27%</td>
<td>63.34%</td>
</tr>
<tr>
<td>20</td>
<td>49.14%</td>
<td>49.82%</td>
<td>49.91%</td>
</tr>
<tr>
<td>40</td>
<td>3.03%</td>
<td>3.62%</td>
<td>3.68%</td>
</tr>
<tr>
<td>60</td>
<td>3.25%</td>
<td>3.83%</td>
<td>3.91%</td>
</tr>
<tr>
<td>80</td>
<td>3.32%</td>
<td>3.91%</td>
<td>3.99%</td>
</tr>
</tbody>
</table>

Figure 306.156: 27NodeBrick circular plate with edge clamped. Displacement error versus Number of side division
306.2.21 Verification of 27NodeBrick circular plate with all edges simply supported

Problem description: Diameter=20m, Height=1m, Force=100N, E=1E8Pa, ν = 0.3.

The four edges are simply supported.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[ D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 N \cdot m \]  

(306.57)

The theoretical solution\(^\text{13}\) is

\[ d = \frac{(5 + \nu)qa^4}{64(1+\nu)D} = \frac{(5 + 0.3) \times 100 N/m^2 \times 10^4 m^4}{64 \times (1 + 0.3) \times 9.1575 \times 10^6 N \cdot m} = 6.956 \times 10^{-3} m \]  

(306.58)

The 27NodeBrick were shown in Figure (306.157) - (306.162).

\[ \text{Figure 306.157: 27NodeBrick edge simply supported circular plate with element side length 10m.} \]

The results were listed in Table (306.43).

The errors were listed in Table (306.44).

The errors were plotted in Figure (306.165).

The Real-ESSI model fei/DSL files for the table above are HERE.

Figure 306.158: 27NodeBrick edge simply supported circular plate with element side length 5m.

Figure 306.159: 27NodeBrick edge simply supported circular plate with element side length 2m.
Figure 306.160: 27NodeBrick edge simply supported circular plate with element side length 1m.

Figure 306.161: 27NodeBrick edge simply supported circular plate with element side length 0.5m.
Figure 306.162: 27NodeBrick edge simply supported circular plate with element side length 0.25m.

Table 306.43: Results for 27NodeBrick circular plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element type</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>2layers</td>
<td>4layers</td>
<td></td>
</tr>
<tr>
<td>Number of diameter divisions</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7.259E-03 m</td>
<td>7.261E-03 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>12</td>
<td>7.083E-03 m</td>
<td>7.084E-03 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>20</td>
<td>7.064E-03 m</td>
<td>7.065E-03 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>40</td>
<td>7.018E-03 m</td>
<td>7.019E-03 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>60</td>
<td>7.029E-03 m</td>
<td>7.030E-03 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>80</td>
<td>7.032E-03 m</td>
<td>7.034E-03 m</td>
<td>6.956E-03 m</td>
</tr>
</tbody>
</table>
Table 306.44: Errors for 27NodeBrick circular plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element type</th>
<th>27NodeBrick</th>
<th>27NodeBrick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>2layers</td>
<td>4layers</td>
</tr>
<tr>
<td>Number of diameter divisions</td>
<td>Height:0.50m</td>
<td>Height:0.25m</td>
</tr>
<tr>
<td>4</td>
<td>4.36%</td>
<td>4.38%</td>
</tr>
<tr>
<td>12</td>
<td>1.82%</td>
<td>1.83%</td>
</tr>
<tr>
<td>20</td>
<td>1.56%</td>
<td>1.57%</td>
</tr>
<tr>
<td>40</td>
<td>0.88%</td>
<td>0.90%</td>
</tr>
<tr>
<td>60</td>
<td>1.04%</td>
<td>1.06%</td>
</tr>
<tr>
<td>80</td>
<td>1.09%</td>
<td>1.11%</td>
</tr>
</tbody>
</table>

Figure 306.163: Error scale 0% - 5%.

Figure 306.164: Error scale 0% - 100%.

Figure 306.165: 27NodeBrick circular plate with edge simply supported Displacement error versus Number of side division.
306.2.22 Verification of 27NodeBrick Finite Element for Boussinesq Problem

306.2.22.1 Introduction

The Boussinesq problem is finding the displacement distribution in the isotropic linearly elastic half-space, subject to a concentrated load applied on the surface and perpendicular to it. The Boussinesq problem diagram is shown in Fig.(306.166).

Boussinesq problem is widely used in geotechnical engineering, especially when designing a foundation which transfers the superstructure load to the soil. To estimate the foundation settlements, it is important to have a reliable numerical solution for the Boussinesq problem.

![Boussinesq problem diagram](Figure Reference: Verruijt, Arnold, and Stefan Van Baars. Soil mechanics. Delft, 2007.)

In 1885, the French scientist Joseph Boussinesq solved the analytic solutions of displacements in the homogeneous isotropic linear elastic half space. In general, the vertical displacement of the surface is

\[ z = 0 : \quad u_z = \frac{P(1 - \nu^2)}{\pi ER} \tag{306.59} \]

where \( P \) is the vertical load, \( \nu \) is the Poisson’s ratio, \( E \) is the elastic modulus, and \( R \) is the distance from the measured point to the loading point.

In this section, the Real-ESSI numerical solution is verified by the analytic solution for the Boussinesq problem.

306.2.22.2 Description of the Verification Model

Since the problem is cylindrical symmetry, a quarter of the entire cube was employed to represent the whole cube. The reduced model was shown in Fig.(306.167).
The side length is 20 meters and the load $P$ is 1N. The elastic modulus $E = 1 \times 10^3 Pa$ and the Poisson’s ratio $\nu = 0.0$.

![Diagram of the reduced model](image1)

Figure 306.167: Reduced model (One quarter model) for the point load on the half space

The boundary conditions are shown in Fig. (306.168) and (306.169).

![Diagram of X-Z view](image2)

Figure 306.168: X-Z view for the reduced model

### 306.2.22.3 Results

**Analytic solution for this model**

According to the previous introduction, the analytic solution on the surface for this problem is

$$z = 0 : \quad u_z = \frac{P(1 - \nu^2)}{\pi ER} = \frac{1}{10^3 \pi \frac{1}{R}}$$  \hspace{1cm} (306.60)

On the face $x = 0$, the distance $R$ on the surface is actually the value of $y$, therefore, the analytic
Figure 306.169: Y-Z view for the reduced model

The solution is

\[ u_z = \frac{1}{10^3 \pi R} = \frac{1}{10^3 \pi y} \tag{306.61} \]

As long as the \( y \) values are substituted, the displacement \( u_z \) is obtained immediately.

**Real-ESSI solution with 27NodeBrickLT**

In Real-ESSI, 27NodeBrickLT elements were used to simulate this model. Each element is \( 2m \times 2m \times 2m \). Since the model is \( 20m \times 20m \times 20m \), the element number is \( 10 \times 10 \times 10 = 1000 \). The vertical displacement at the surface was recorded.

Since the model is symmetric, when the results were plotted, the other half results were achieved by symmetry.

**Comparison between the analytic and 27NodeBrickLT solution**

The Real-ESSI and analytic results were plotted in Fig. (306.170). Note that the analytic solution for location \( y = 0 \) is infinity, which was not plotted in the figure below.

**306.2.22.4 Error Analysis**

1. Mismatch at the loading point.

First of all, when \( x = 0 \) at the loading point, the analytic solution is infinite. From the perspective of practical engineering, this analytic solution is flawed because the displacement cannot be infinite.

The infinite solution is due to the elastic assumption. In consideration of the plasticity, the analytic solution will not be infinite.

In Real-ESSI, the displacement at the loading point is not infinite because the infinite value is averaged by the integration during the finite element calculation. Also, at the loading point,
27NodeBrickLT has a much larger displacement than that of 8NodeBrickLT. This is because 27NodeBrickLT has a relatively denser mesh than 8NodeBrickLT. So the maximum value at the loading point is higher than that of 8NodeBrickLT.

2. Mismatch at other locations.

Except at the loading point, the analytic solution is not exactly equal to the numerical solution at other locations. This is because the verification example employs a simplified bounded cube to represent unbounded half space. The original analytic solution is for the half space, which is not true for the verification model. Not only the horizontal space but also the bottom space are removed from the model. This means the analytic solution is not perfect for this bounded cube. However, since the cube is very great, the analytic solution is similar to the Real-ESSI numerical solution. In addition, the brick elements are also verified by other models, like beam, plate and shells.

306.3 Verification of Dynamic, Single Phase Solid Modeling and Simulation
Chapter 307

Verification and Validation for Static and Dynamic Behavior of Structural Elements

(In collaboration with Prof. José Abell, Dr. Yuan Feng, and Dr. Han Yang)
307.1 Chapter Summary and Highlights

307.2 Verification of Static, Beam-Column Finite Element Modeling and Simulation

307.3 Bernoulli Beam Elements with 12DOFs and 9DOFs

Figures 307.1 and 307.2

307.3.1 FEM Model

![Finite element model for static analysis](image1)

Figure 307.1: Finite element model for static analysis

![Finite element model for dynamic analysis](image2)

Figure 307.2: Finite element model for dynamic analysis

307.3.2 Static Analysis

Figures 307.3
\[ E = 5Pa \]
\[ I = 16m^4 \]
\[ L = 2m \]
\[ P = 1N \]
\[ \Delta = \frac{PL^3}{3EI} = 0.033m \]

Figure 307.3: Comparison of static displacements (pseudo time) of the top nodes, Force time history applied to the top node

307.3.3 Dynamic Analysis

Figures 307.4 and 307.5
Figure 307.4: Comparison of displacement time histories of the top nodes, Displacement time history applied to the node with 6DOF, mass comes from beam density

Figure 307.5: Comparison of free vibration displacement time histories of the top nodes, mass comes from beam density
307.3.4 Bernoulli Beam, Comparison of Eigen Frequencies

Table 307.1

Table 307.1: Comparison of eigen frequencies between models of using 9DOF beam and 12DOF beam

<table>
<thead>
<tr>
<th>Mode</th>
<th>9DOF</th>
<th>12DOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.264559</td>
<td>0.264236</td>
</tr>
<tr>
<td>2</td>
<td>0.268474</td>
<td>0.268064</td>
</tr>
<tr>
<td>3</td>
<td>0.308202</td>
<td>0.308202</td>
</tr>
</tbody>
</table>
307.4 Timoshenko Beam

Models used:

- Model with 27-node-brick model is used as the benchmark. The model is $10\text{m} \times 10\text{m} \times 60\text{m}$, each element is $2\text{m} \times 2\text{m} \times 2\text{m}$, so there are $5 \times 5 \times 30 = 750$ elements in total.

- Model with 5 Timoshenko beams are used to test the performance of the Timoshenko element. Material properties and cross-section properties are kept the same as those of the brick model. Various values of the shear correction factor are tested.

- Model with 5 Bernoulli beam elements is also tested.

- For both beam models, consistent mass is used.

Input files for all the models shown are available [HERE](#).

Figures of models are shown below:

![Figure 307.6](#)  
**Figure 307.6:** Cantilever model made of 27-node-brick elements.
Figure 307.7: Cantilever model made of multiple Timoshenko or Bernoulli elements.
Eigenanalysis results are shown in the following table and figures.

Table 307.2: Comparison of eigenfrequencies for 27-node-brick, Timoshenko beam with different shear correction factors, and Bernoulli beam models.

<table>
<thead>
<tr>
<th>Eigenmode</th>
<th>Eigenfrequency (Hz)</th>
<th>27-Node-Brick</th>
<th>Timoshenko (shear factor=1)</th>
<th>Timoshenko (shear factor=2)</th>
<th>Timoshenko (shear factor=100)</th>
<th>Bernoulli</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>27-Node-Brick</td>
<td>Timoshenko (shear factor=1)</td>
<td>Timoshenko (shear factor=2)</td>
<td>Timoshenko (shear factor=100)</td>
<td>Bernoulli</td>
</tr>
<tr>
<td>Bending (1st)</td>
<td></td>
<td>1.63364</td>
<td>1.63375</td>
<td>1.64358</td>
<td>1.65336</td>
<td>1.65356</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.00000</td>
<td>2.00000</td>
<td>2.00000</td>
<td>2.00000</td>
<td>2.00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.00000</td>
<td>4.00000</td>
<td>4.00000</td>
<td>4.00000</td>
<td>4.00000</td>
</tr>
<tr>
<td>Torsion (1st)</td>
<td></td>
<td>9.30771</td>
<td>10.1473</td>
<td>10.1473</td>
<td>10.1473</td>
<td>10.1473</td>
</tr>
<tr>
<td>Axial (1st)</td>
<td></td>
<td>15.4575</td>
<td>15.5002</td>
<td>15.5002</td>
<td>15.5002</td>
<td>15.5002</td>
</tr>
<tr>
<td>Bending (3rd)</td>
<td></td>
<td>22.8703</td>
<td>23.7943</td>
<td>25.246</td>
<td>26.9177</td>
<td>26.9542</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.00000</td>
<td>8.00000</td>
<td>8.00000</td>
<td>8.00000</td>
<td>8.00000</td>
</tr>
<tr>
<td>Torsion (2nd)</td>
<td></td>
<td>27.9199</td>
<td>31.4476</td>
<td>31.4476</td>
<td>31.4476</td>
<td>31.4476</td>
</tr>
</tbody>
</table>

Figure 307.8: First 9 eigenmodes of the cantilever model made of 27-node-brick elements.
Figure 307.9: First 9 eigenmodes of the cantilever model made of Timoshenko beam elements.
307.5 Verification of Shell (Felippa-ANDES) Finite Element Modeling and Simulation

The verification and validation of the behavior of the ANDES (Assumed Natural Deviatoric Strain) shell finite element implemented in essi is described in this section. The verification is split up into several cases intended to test different aspects of the implementation. The tests are based on well-known closed form solutions to elasticity problems which can be modelled using shells. Further, the verification is divided into static and dynamic tests.

307.5.1 Static Tests

The purpose of the static tests is to verify that the stiffness matrices generated by the 4 Node ANDES Shell are useful to approximate well known cases of solutions to beam theory.

Tests are done to verify the bending component and the membrane component independently, because the behaviour of both is assumed de-coupled in this linear implementation.

307.5.1.1 Bending Component Verification

These tests compare the bending performance of a simple mesh of ANDES shells for the case of simple (Bernoulli) beam examples.

Test 1: Concentrated transversal tip load

The test file for this case can be found in

(ESSI SOURCE FOLDER)/Verification_Examples/...
.../Static_and_DYNAMIC_Behavior_of_Structural_Elements/...
.../Shell_ANDES/static/Test_shell_andes_1_bending_transverse.fei

For a prismatic beam made of a homogeneous elastic isotropic material of modulus of elasticity $E$, cross section $I$ and length $L$ we have that, under Bernoulli-Euler theory, the displacement at the tip due to a concentrated load $P$ is

$$\delta_{\text{tip}} = \frac{PL^3}{3EI}$$

and for this test case's geometry

$$\delta_{\text{tip}} = \frac{qBL^3}{3EBh^3/12} = \frac{4qL^3}{Eh^3}$$

With the purpose of measuring the errors as percent agreement with the theoretical results, the load $q$ is computed such that the tip displacement is 100 units. Thus, in terms of the geometry of the problem.

$$q = 100 \times \frac{Eh^3}{4L^5}$$

This force is translated into nodal equivalent forces by applying half the total load $qB$ to each tip node in the vertical direction and fixing the $x$ direction rotational degrees of freedom for the tip nodes.

The parameters chosen for this case are

1. $h = 1*m$; // Shell thickness
2. $E_{\text{shell}} = 1*N/m^2$; // Elastic modulus
3. $\nu = 0.0$; // Poisson’s ratio
4. $L = 1*m$; // Beam length
5. $B = 0.2*m$; // Beam width
6. $\rho = 0*kg/m^3$; // Mass density

For different number of subdivisions, here are the results of the tip displacement.

<table>
<thead>
<tr>
<th>$N_{\text{subd}}$</th>
<th>$u_z$</th>
<th>$N_{\text{subd}}$</th>
<th>$u_z$</th>
<th>$N_{\text{subd}}$</th>
<th>$u_z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>96.2118</td>
<td>42</td>
<td>100.009</td>
<td>83</td>
<td>100.002</td>
</tr>
<tr>
<td>7</td>
<td>100.096</td>
<td>48</td>
<td>100.007</td>
<td>89</td>
<td>100.002</td>
</tr>
<tr>
<td>13</td>
<td>100.068</td>
<td>54</td>
<td>100.006</td>
<td>95</td>
<td>100.002</td>
</tr>
<tr>
<td>19</td>
<td>100.039</td>
<td>60</td>
<td>100.005</td>
<td>101</td>
<td>100.002</td>
</tr>
<tr>
<td>25</td>
<td>100.024</td>
<td>66</td>
<td>100.004</td>
<td>107</td>
<td>100.001</td>
</tr>
<tr>
<td>31</td>
<td>100.016</td>
<td>72</td>
<td>100.003</td>
<td>112</td>
<td>100.001</td>
</tr>
<tr>
<td>37</td>
<td>100.012</td>
<td>77</td>
<td>100.003</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The element displays sensitivity to the aspect ratio of its sides. Indeed, choosing a different set of parameters
h = 1*m; // Shell thickness
Eshell = 1*N/m^2; // Elastic modulus
nu = 0.0; // Poisson's ratio
L = 1*m; // Beam length
B = 0.2*m; // Beam width
rho = 0*kg/m^3; // Mass density

yields slightly different results

<table>
<thead>
<tr>
<th>N_{subd}</th>
<th>u_z</th>
<th>N_{subd}</th>
<th>u_z</th>
<th>N_{subd}</th>
<th>u_z</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>93.7897</td>
<td>42</td>
<td>99.9958</td>
<td>83</td>
<td>100.001</td>
</tr>
<tr>
<td>7</td>
<td>99.5267</td>
<td>48</td>
<td>99.998</td>
<td>89</td>
<td>100.001</td>
</tr>
<tr>
<td>13</td>
<td>99.8587</td>
<td>54</td>
<td>99.9993</td>
<td>95</td>
<td>100.001</td>
</tr>
<tr>
<td>19</td>
<td>99.9409</td>
<td>60</td>
<td>100.01</td>
<td>101</td>
<td>100.001</td>
</tr>
<tr>
<td>25</td>
<td>99.9713</td>
<td>66</td>
<td>100.001</td>
<td>107</td>
<td>100.001</td>
</tr>
<tr>
<td>31</td>
<td>99.985</td>
<td>72</td>
<td>100.001</td>
<td>112</td>
<td>100.001</td>
</tr>
<tr>
<td>37</td>
<td>99.992</td>
<td>77</td>
<td>100.001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Test 2: Concentrated tip moment

The test file for this case can be found in
(ESSI SOURCE FOLDER)/Verification_Examples/...
.../Static_and_Dynamic_Behavior_of_Structural.Elements/...
.../Shell_ANDES/static/Test_shell_andes_2_bending_transverse.fei

Analogous to the previous test, for a cantilever beam with a tip moment the tip rotation is

\[ \theta_{\text{tip}} = \frac{ML}{EI} \]

and for this test case’s geometry

\[ \theta_{\text{tip}} = \frac{mBL}{EBh^3/12} = \frac{12mL}{Eh^3} \]

In this case, the parameters are selected such that this tip rotation is of 100 units. Execution of the test case shows that with \( N_{\text{subd}} = 2 \) the theoretical value is met with 100% accuracy. This is because the moment field is constant inside the beam and the elements are capable of reproducing this field with accuracy.

Test 3: Concentrated mid-span transversal load
This case is similar to test case 1. The accuracy for 2 elements is 96.2% agreement with the theoretical solution. For $N_{\text{subd}} = 4$ (4 elements total) the accuracy climbs to 99.7%. The test file for this case can be found in

(ESSI SOURCE FOLDER)/Verification_Examples/

.../Static_and_Dynamic_Behavior_of_Structural_Elements/

.../Shell_ANDES/static/Test_shell_andes_3_bending.transverse.fei

Test 4: Transversal distributed load

The distributed load is generated by accelerating the beam transversally with an acceleration which produces a 100 (unit) displacement. This provides an indirect test to the mass matrix which will be further tested in dynamic tests. The uniformly distributed load $q$ which produces the $\delta_{\text{tip}}$ unit displacement and the corresponding tip rotation are

$$q = \frac{EI}{L^4} \delta_{\text{tip}} \quad \text{and} \quad \theta_{y\text{tip}} = \frac{4}{3L} \delta_{\text{tip}}$$

from this value, the required acceleration is computed as

$$a = \frac{q}{\rho BH}$$

where $\rho$ is the unit-weight of the material used to compute the mass matrix.

For two subdivisions (6 nodes) the results of the two nodes located at the tip of the beam are

<table>
<thead>
<tr>
<th>Node</th>
<th>ux</th>
<th>rx</th>
<th>uy</th>
<th>ry</th>
<th>uz</th>
<th>rz</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.000000</td>
<td>30.550500</td>
<td>0.000000</td>
<td>-133.333000</td>
<td>102.541000</td>
<td>0.000000</td>
</tr>
<tr>
<td>6</td>
<td>0.000000</td>
<td>-30.550500</td>
<td>0.000000</td>
<td>-133.333000</td>
<td>102.541000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

First, a 2.5% accuracy is reached in the tip displacement. Second, it is noteworthy to mention that the current formulation of the mass matrix will produce rotations around the $x$ axis for the nodes. One reason for this is that the chosen mass matrix is not consistent with the stiffness matrix instead, it is borrowed from a similar element. The reason behind this decision is the fact that the very accurate ANDES stiffness formulation lacks a displacement interpolation scheme which is necessary to produce a

---

1. The First ANDES Elements: 9-DOF Plate Bending Triangles Carmello Militello & Carlos A. Felippa
3. Chapter 32 of Felippa’s Lecture Notes Finite element templates for bending
consistent mass matrix. Furthermore, restraining these rotation degrees of freedom does not lead to an improved solution.

For 4 subdivisions, the results at the tip are:

<table>
<thead>
<tr>
<th>Node</th>
<th>ux</th>
<th>rx</th>
<th>uy</th>
<th>ry</th>
<th>uz</th>
<th>rz</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0.00000</td>
<td>31.67100</td>
<td>0.00000</td>
<td>-133.33300</td>
<td>100.89100</td>
<td>0.00000</td>
</tr>
<tr>
<td>10</td>
<td>0.00000</td>
<td>-31.67100</td>
<td>0.00000</td>
<td>-133.33300</td>
<td>100.89100</td>
<td>0.00000</td>
</tr>
</tbody>
</table>

and for 20.

<table>
<thead>
<tr>
<th>Node</th>
<th>ux</th>
<th>rx</th>
<th>uy</th>
<th>ry</th>
<th>uz</th>
<th>rz</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>0.00000</td>
<td>32.09580</td>
<td>0.00000</td>
<td>-133.33300</td>
<td>100.03900</td>
<td>0.00000</td>
</tr>
<tr>
<td>42</td>
<td>0.00000</td>
<td>-32.09580</td>
<td>0.00000</td>
<td>-133.33300</td>
<td>100.03900</td>
<td>0.00000</td>
</tr>
</tbody>
</table>

and back to 2 subdivisions but this time with an aspect ratio $L/B = 2$, by changing $B$ so that the expected tip displacement remains the same, we get

<table>
<thead>
<tr>
<th>Node</th>
<th>ux</th>
<th>rx</th>
<th>uy</th>
<th>ry</th>
<th>uz</th>
<th>rz</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.00000</td>
<td>3.89020</td>
<td>0.00000</td>
<td>-133.33300</td>
<td>99.81010</td>
<td>0.00000</td>
</tr>
<tr>
<td>6</td>
<td>0.00000</td>
<td>-3.89020</td>
<td>0.00000</td>
<td>-133.33300</td>
<td>99.81010</td>
<td>0.00000</td>
</tr>
</tbody>
</table>

which shows that the effect of the inconsistent mass matrix is ameliorated for elements with a better aspect ratio. The elements in this case are square in shape since $L/B = 2$ and $N_{subd} = 2$.

In conclusion increasing accuracy for bending problems not only involves making the elements smaller but also improving their aspect ratio. This is especially critical in dynamic problems (or self weight problems). The test file for this case can be found in

(ESSI SOURCE FOLDER)/Verification_Examples/...
.../Static_and_Dynamic_Behavior_of_Structural_Elements/...
.../Shell_ANDES/static/Test_shell_anes4_bending_transverse.fei
307.5.1.2 Membrane Component Verification

These tests compare the performance of the membrane component of the ANDES Shell by modelling simple beams and comparing the approximation with Bernoulli-beam theoretical results.

Test 5: Longitudinal tip load

The parameters used to test this case are,

```plaintext
1 h = 0.2*m; // Shell thickness
2 Eshell = 1*N/m^2; // Elastic modulus
3 nu = 0.0; // Poisson's ratio
4 L = 30*m; // Beam length
5 B = 3*m; // Beam width
6 rho = 0.0*kg/m^3; // Mass density
7 delta_tip = 100*m; // Target tip displacement
8
9 Nsubd = 2; // Number of side subdivisions
```

the tip load was computed to give $\delta_{\text{tip}} = 100$ from

$$F_{\text{tip}} = \frac{AE}{L}\delta_{\text{tip}}$$

Additionally, the tip rotation degrees of freedom about $z$ axis ($rz$) were fixed to enforce the uniform loading condition. The results for 2 subdivisions observed at the two tip nodes is

```plaintext
1 Node : 5
2 ux = 100.000000, rx = 0.000000
3 uy = -0.000000, ry = 0.000000
4 uz = 0.000000, rz = 0.000000
5
6 Node : 6
7 ux = 100.000000, rx = 0.000000
8 uy = -0.000000, ry = 0.000000
9 uz = 0.000000, rz = 0.000000
```
Test 6: Longitudinal self-weight load

As in test 4, the distributed load for this test was generated by using an acceleration field with a magnitude such that it generates 100 units of displacement in DOF $ux$. The required body force $q$ (per unit length) is derived from linear elasticity to be

$$q = \frac{2EA}{L^2} \delta_{\text{tip}}$$

from which the acceleration is found to be given by $a = \frac{q}{\rho BH}$. The tip nodes were fixed to move only in the $x$ direction. The model parameters used for this test are,

1. $h = 1*m$; // Shell thickness
2. $E_{\text{shell}} = 1*N/m^2$; // Elastic modulus
3. $\nu = 0.0$; // Poisson's ratio
4. $L = 1*m$; // Beam length
5. $B = 1*m$; // Beam width
6. $\rho = 100*kg/m^3$; // Mass density
7. $\delta_{\text{tip}} = 100*m$; // Target tip displacement
8. $N_{\text{subd}} = 2$; // Number of side subdivisions

and the tip displacements

1. Node : 5
   2. $ux = 100.000000$, $rx = 0.000000$
   3. $uy = 0.000000$, $ry = 0.000000$
   4. $uz = 0.000000$, $rz = 0.000000$
5. Node : 6
   6. $ux = 100.000000$, $rx = 0.000000$
   7. $uy = 0.000000$, $ry = 0.000000$
   8. $uz = 0.000000$, $rz = 0.000000$

Which means that in the quadratic displacement field coming from a uniform external load can be captured exactly by this element.

The test file for this case can be found in (ESSI SOURCE FOLDER)/Verification_Examples/.../Static_and_Dynamic_Behavior_of_Structural_Elements/.../Shell_ANDES/static/Test_shell_andes_5_membrane_axial.fei

Test 7: Transversal tip load

This test is identical to test 1, except the beam mesh is placed sideways so that the membrane component is used instead of the bending one. Since the membrane part can capture deformation due to shear, the ratio $L/B$ is set to 10 so that this does not affect the results.
h = 0.2*m; // Shell thickness
Eshell = 1*N/m^2; // Elastic modulus
nu = 0.0; // Poisson's ratio
L = 10*m; // Beam length
B = 1.0*m; // Beam height
rho = 0*kg/m^3; // Mass density
d_tip = 100*m; // Target tip displacement
Nsubd = 10; // Number of side subdivisions

Results at the tip show less than 1% error with theoretical results when shear component is made small.

Node : 21
ux = -7.513920, rx = 0.000000
uy = -100.650000, ry = 0.000000
uz = 0.000000, rz = -15.072400

Node : 22
ux = 7.513920, rx = 0.000000
uy = -100.650000, ry = 0.000000
uz = 0.000000, rz = -15.072400

It would be interesting to test this component vs. a beam theory which incorporates deformation due to shear such as Timoshenko beams.

The test file for this case can be found in (ESSI SOURCE FOLDER)/Verification_Examples/.../Static_and_Dynamic_Behavior_of_Structural.Elements/.../Shell_ANDES/static/Test_shell_andes_7_membrane_transverse.fei

Test 8: Transversal self-weight load This test is analogous to test 4, except the membrane component is being tested instead of the bending. Again, the L/B ratio is kept at 10 to avoid shear deformation creeping into the results perceptively.

h = 1*m; // Shell thickness
Eshell = 1*N/m^2; // Elastic modulus
nu = 0.0; // Poisson's ratio
L = 10*m; // Beam length
B = 0.5*m; // Beam width
rho = 100*kg/m^3; // Mass density
delta_tip = 100*m; // Target tip displacement
Nsubd = 10; // Number of side subdivisions

Displacement results at the tip nodes show less than 1% error when compared to Bernoulli beam theory.

Node : 21
ux = 3.325110, rx = 0.000000
uy = 99.780900, ry = 0.000000
uz = 0.000000, rz = 13.301200
Node : 22
ux = -3.325110, rx = 0.000000
uy = 99.780900, ry = 0.000000
uz = 0.000000, rz = 13.301200

The test file for this case can be found in (ESSI SOURCE FOLDER)/Verification_Examples/...
.../Static_and_Dynamic_Behavior_of_Structural_Elements/...
.../Shell_ANDES/static/Test_shell_andes_8_membrane_transverse.fei
307.5.2 Dynamic Tests

The purpose of the dynamic tests is to verify that the mass matrix adopted for these elements (which is not consistent with the stiffness) is adequate for Dynamic analysis. Also tested herein are the geometric transformations.

Again, the tests are divided into bending and membrane components which are tested independently. Also, an eigenvalue analysis is performed to verify accordance with theoretical results for continuous Bernoulli beams. Finally, geometric transformations are tested by performing an eigenvalue analysis for the same (unrestrained) beam in different orientations (pitch, yaw and roll) which should have invariant eigenvalues.

307.5.2.1 Bending Component

![Mode 1, T = 0.999959s](image1)
![Mode 2, T = 0.159539s](image2)
![Mode 3, T = 0.0858275s](image3)
![Mode 4, T = 0.0569888s](image4)
![Mode 5, T = 0.0291146s](image5)
![Mode 6, T = 0.027886s](image6)
307.5.2.2 Membrane Component

Mode 1, $T = 0.998022s$

Mode 2, $T = 0.15862s$

Mode 3, $T = 0.0564893s$

Mode 4, $T = 0.0287508s$

Mode 5, $T = 0.0173475s$

Mode 6, $T = 0.0129099s$
307.5.2.3 Geometric Transformations
307.6 Verification of 4NodeANDES elements

307.6.1 Verification of 4NodeANDES cantilever beams

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.0$. Use the shear deformation coefficient $\kappa = 1.2$. The force direction was shown in Figure (307.10).

![Figure 307.10: Problem description for cantilever beams.](image)

Theoretical displacement (bending and shear deformation):

$$d = \frac{FL^3}{3EI} + \frac{FL}{GA_v}$$

$$= \frac{FL^3}{3E\frac{bh^3}{12}} + \frac{FL}{\frac{E}{2(1+\nu)} \frac{bh}{k}}$$

$$= \frac{100N \times 6^3m^3}{3 \times 10^8N/m^2 \times \frac{1}{12}m^4} + \frac{100N \times 6m}{\frac{10}{2} \times 10^7N/m^2 \times 1m^2 \times \frac{5}{6}}$$

$$= 8.64 \times 10^{-4}m + 0.144 \times 10^{-4}m$$

$$= 8.784 \times 10^{-4}m$$

(307.1)

4NodeANDES element model:

- **Force direction: perpendicular to plane (bending)**

When the force direction is perpendicular to the plane, only the bending deformation is calculated in 4NodeANDES elements.

The 4NodeANDES elements were shown in Figure (307.14).

- **Force direction: inplane force**

When the force direction is inplane, both the bending and shear deformation are calculated in 4NodeANDES elements.
The 4NodeANDES elements under inplane force were shown in Figure (307.18).

The Real-ESSI results for the force perpendicular to plane (bending) were listed in Table (307.3). The theoretical solution is 8.784E-04 m. The Real-ESSI results for the inplane force were listed in Table 307.3: Results for 4NodeANDES cantilever beams under the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Element number</th>
<th>1</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4NodeANDES</td>
<td>6.56E-04 m</td>
<td>8.27E-04 m</td>
<td>8.86E-04 m</td>
</tr>
<tr>
<td>Error</td>
<td>25.34%</td>
<td>5.87%</td>
<td>0.83%</td>
</tr>
</tbody>
</table>

Table (307.4).

The theoretical solution is 8.784E-04 m.

The errors were plotted in Figure (307.21).

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 307.15: One 4NodeANDES element.

Figure 307.16: Two 4NodeANDES elements.

Figure 307.17: Six 4NodeANDES elements.

Figure 307.18: 4NodeANDES elements for cantilever beams under inplane force.

Table 307.4: Results for 4NodeANDES cantilever beams under the inplane force.

<table>
<thead>
<tr>
<th>Element number</th>
<th>1</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4NodeANDES</td>
<td>6.70E-04 m</td>
<td>8.27E-04 m</td>
<td>8.64E-04 m</td>
</tr>
<tr>
<td>Error</td>
<td>23.77%</td>
<td>5.89%</td>
<td>1.65%</td>
</tr>
</tbody>
</table>
Figure 307.19: Error scale 0% - 30%.

Figure 307.20: Error scale 0% - 100%.

Figure 307.21: 4NodeANDES cantilever beam for different element number. Displacement error versus number of elements.
307.6.2 Verification of 4NodeANDES cantilever beam for different Poisson’s ratio

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 - 0.49 \). The force direction was shown in Figure (307.22).

![Figure 307.22: Problem description for cantilever beams of different Poisson’s ratios.](image)

The theoretical solution for \( \nu = 0.0 \) was calculated below, while the solution for other Poisson’s ratio were calculated by the similar process.

Theoretical displacement (bending and shear deformation):

\[
d = \frac{FL^3}{3EI} + \frac{FL}{GA_v}
= \frac{FL^3}{3E\frac{bh^3}{12}} + \frac{FL}{\frac{E}{2(1+\nu)} \frac{bh}{c}}
= \frac{100N \times 6^3 m^3}{3 \times 10^8 N/m^2 \times \frac{1}{12} m^4} + \frac{100N \times 6m}{\frac{10}{2} \times 10^7 N/m^2 \times 1m^2 \times \frac{5}{6}}
= 8.64 \times 10^{-4} m + 0.144 \times 10^{-4} m
= 8.784 \times 10^{-4} m
\]  

(307.2)

The rotation angle at the end:

\[
\theta = \frac{FL^2}{2EI} = \frac{100N \times 6^2 m^2}{2 \times 10^8 N/m^2 \times \frac{1}{12} m^4} = 2.16 \times 10^{-4} \text{ rad} = 0.0124^\circ
\]  

(307.3)

The 4NodeANDES elements for cantilever beams of different Poisson’s ratios were shown in Figure (307.23) and (307.24):

The Real-ESSI results for the force **perpendicular to plane (bending)** were listed in Table (307.5) - (307.7).

The errors were plotted in Figure (307.27).

The Real-ESSI results for the **inplane force** were listed in Table (307.8) - (307.10).
Figure 307.23: 4NodeANDES elements for different Poisson’s ratios under the force perpendicular to plane (bending).

Figure 307.24: 4NodeANDES elements for different Poisson’s ratios under the inplane force.

Table 307.5:  **Displacement error** results for 4NodeANDES with element side length 1 m under the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement (all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.639E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>1.38%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.635E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>1.49%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.622E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>1.71%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.599E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>2.04%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.566E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>2.48%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.522E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>3.05%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.466E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>3.75%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.398E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>4.59%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.315E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>5.60%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.216E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>6.78%</td>
</tr>
<tr>
<td>0.49</td>
<td>8.124E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>7.88%</td>
</tr>
</tbody>
</table>

The errors were plotted in Figure (307.27).

The angle results for the force **perpendicular to plane (bending)** were listed in Table (307.11). The errors were plotted in Figure (307.33).
Table 307.6: **Displacement error** results for 4NodeANDES with element side length 0.5 m under the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.724E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>0.68%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.724E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>0.76%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.717E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>0.93%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.703E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>1.17%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.682E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>1.49%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.652E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>1.91%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.615E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>2.42%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.569E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>3.02%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.514E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>3.73%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.449E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>4.54%</td>
</tr>
<tr>
<td>0.49</td>
<td>8.388E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>5.30%</td>
</tr>
</tbody>
</table>

Table 307.7: **Displacement error** results for 4NodeANDES with element side length 0.25 m under the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.640E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>1.64%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.637E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>1.75%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.627E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>1.95%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.611E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>2.21%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.588E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>2.56%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.559E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>2.97%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.523E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>3.46%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.480E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>4.03%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.429E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>4.69%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.370E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>5.44%</td>
</tr>
<tr>
<td>0.49</td>
<td>8.316E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>6.11%</td>
</tr>
</tbody>
</table>
Figure 307.25: Error scale 0% - 15%.

Figure 307.26: Error scale 0% - 100%.

Figure 307.27: 4NodeANDES cantilever beam for force perpendicular to the plane (bending). Displacement error versus Poisson’s ratio

The Real-ESSI results for the inplane force were listed in Table (307.14 - (307.16). The errors were plotted in Figure (307.33). The Real-ESSI model fei/DSL files for the table above are HERE.
Table 307.8: **Displacement error** results for 4NodeANDES with element side length 1 m under the inplane force.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.790E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>0.07%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.799E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>0.09%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.809E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>0.12%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.821E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>0.17%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.835E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>0.25%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.853E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>0.37%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.878E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>0.56%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.913E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>0.87%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.971E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>1.44%</td>
</tr>
<tr>
<td>0.45</td>
<td>9.107E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>2.89%</td>
</tr>
<tr>
<td>0.49</td>
<td>9.901E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>11.79%</td>
</tr>
</tbody>
</table>

Table 307.9: **Displacement error** results for 4NodeANDES with element side length 0.5 m under the inplane force.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.784E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>0.00%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.788E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>0.04%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.787E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>0.13%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.782E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>0.27%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.772E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>0.47%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.759E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>0.70%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.742E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>0.98%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.722E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>1.29%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.699E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>1.63%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.679E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>1.94%</td>
</tr>
<tr>
<td>0.49</td>
<td>8.709E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>1.67%</td>
</tr>
</tbody>
</table>
Table 307.10: **Displacement error** results for 4NodeANDES with element side length 0.25 m under the inplane force.

<table>
<thead>
<tr>
<th>Poisson's ratio</th>
<th>4NodeANDES displacement</th>
<th>Theory displacement (bending)</th>
<th>Theory displacement (shear)</th>
<th>Theory displacement(all)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>8.782E-04 m</td>
<td>8.640E-04 m</td>
<td>1.440E-05 m</td>
<td>8.784E-04 m</td>
<td>0.02%</td>
</tr>
<tr>
<td>0.05</td>
<td>8.786E-04 m</td>
<td>8.640E-04 m</td>
<td>1.512E-05 m</td>
<td>8.791E-04 m</td>
<td>0.06%</td>
</tr>
<tr>
<td>0.10</td>
<td>8.788E-04 m</td>
<td>8.640E-04 m</td>
<td>1.586E-05 m</td>
<td>8.799E-04 m</td>
<td>0.12%</td>
</tr>
<tr>
<td>0.15</td>
<td>8.786E-04 m</td>
<td>8.640E-04 m</td>
<td>1.659E-05 m</td>
<td>8.806E-04 m</td>
<td>0.23%</td>
</tr>
<tr>
<td>0.20</td>
<td>8.781E-04 m</td>
<td>8.640E-04 m</td>
<td>1.734E-05 m</td>
<td>8.813E-04 m</td>
<td>0.37%</td>
</tr>
<tr>
<td>0.25</td>
<td>8.774E-04 m</td>
<td>8.640E-04 m</td>
<td>1.808E-05 m</td>
<td>8.821E-04 m</td>
<td>0.53%</td>
</tr>
<tr>
<td>0.30</td>
<td>8.763E-04 m</td>
<td>8.640E-04 m</td>
<td>1.884E-05 m</td>
<td>8.828E-04 m</td>
<td>0.74%</td>
</tr>
<tr>
<td>0.35</td>
<td>8.750E-04 m</td>
<td>8.640E-04 m</td>
<td>1.959E-05 m</td>
<td>8.836E-04 m</td>
<td>0.97%</td>
</tr>
<tr>
<td>0.40</td>
<td>8.734E-04 m</td>
<td>8.640E-04 m</td>
<td>2.035E-05 m</td>
<td>8.844E-04 m</td>
<td>1.24%</td>
</tr>
<tr>
<td>0.45</td>
<td>8.717E-04 m</td>
<td>8.640E-04 m</td>
<td>2.111E-05 m</td>
<td>8.851E-04 m</td>
<td>1.52%</td>
</tr>
<tr>
<td>0.49</td>
<td>8.706E-04 m</td>
<td>8.640E-04 m</td>
<td>2.173E-05 m</td>
<td>8.857E-04 m</td>
<td>1.71%</td>
</tr>
</tbody>
</table>

Table 307.11: **Rotation angle** results for element side length 1 m under the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Poisson's ratio</th>
<th>4NodeANDES angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.238E-02</td>
<td>1.240E-02</td>
<td>0.19%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.237E-02</td>
<td>1.240E-02</td>
<td>0.23%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.236E-02</td>
<td>1.240E-02</td>
<td>0.34%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.234E-02</td>
<td>1.240E-02</td>
<td>0.52%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.230E-02</td>
<td>1.240E-02</td>
<td>0.78%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.226E-02</td>
<td>1.240E-02</td>
<td>1.12%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.221E-02</td>
<td>1.240E-02</td>
<td>1.54%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.214E-02</td>
<td>1.240E-02</td>
<td>2.07%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.206E-02</td>
<td>1.240E-02</td>
<td>2.70%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.197E-02</td>
<td>1.240E-02</td>
<td>3.46%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.188E-02</td>
<td>1.240E-02</td>
<td>4.16%</td>
</tr>
</tbody>
</table>
Figure 307.28: Error scale 0% - 10%.

Figure 307.29: Error scale 0% - 100%.

Figure 307.30: 4NodeANDES cantilever beam for inplane force. **Displacement error** versus Poisson's ratio.
Table 307.12: **Rotation angle** results for element side length 0.5 m the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.239E-02</td>
<td>1.240E-02</td>
<td>0.10%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.238E-02</td>
<td>1.240E-02</td>
<td>0.13%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.237E-02</td>
<td>1.240E-02</td>
<td>0.22%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.236E-02</td>
<td>1.240E-02</td>
<td>0.36%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.233E-02</td>
<td>1.240E-02</td>
<td>0.55%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.230E-02</td>
<td>1.240E-02</td>
<td>0.81%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.226E-02</td>
<td>1.240E-02</td>
<td>1.13%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.221E-02</td>
<td>1.240E-02</td>
<td>1.52%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.216E-02</td>
<td>1.240E-02</td>
<td>1.97%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.209E-02</td>
<td>1.240E-02</td>
<td>2.51%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.203E-02</td>
<td>1.240E-02</td>
<td>3.00%</td>
</tr>
</tbody>
</table>

Table 307.13: **Rotation angle** results for element side length 0.25 m under the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.238E-02</td>
<td>1.240E-02</td>
<td>0.19%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.237E-02</td>
<td>1.240E-02</td>
<td>0.21%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.237E-02</td>
<td>1.240E-02</td>
<td>0.28%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.235E-02</td>
<td>1.240E-02</td>
<td>0.39%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.233E-02</td>
<td>1.240E-02</td>
<td>0.56%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.230E-02</td>
<td>1.240E-02</td>
<td>0.78%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.227E-02</td>
<td>1.240E-02</td>
<td>1.05%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.223E-02</td>
<td>1.240E-02</td>
<td>1.38%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.218E-02</td>
<td>1.240E-02</td>
<td>1.77%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.212E-02</td>
<td>1.240E-02</td>
<td>2.23%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.207E-02</td>
<td>1.240E-02</td>
<td>2.64%</td>
</tr>
</tbody>
</table>
Figure 307.31: Error scale 0% - 5%.

Figure 307.32: Error scale 0% - 100%.

Figure 307.33: 4NodeANDES cantilever beam for force perpendicular to the plane (bending).

Rotation angle error versus Poisson's ratio
Table 307.14: Rotation angle results for element side length 1 m under the inplane force.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.254E-02</td>
<td>1.240E-02</td>
<td>1.14%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.255E-02</td>
<td>1.240E-02</td>
<td>1.19%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.256E-02</td>
<td>1.240E-02</td>
<td>1.26%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.257E-02</td>
<td>1.240E-02</td>
<td>1.35%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.258E-02</td>
<td>1.240E-02</td>
<td>1.47%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.260E-02</td>
<td>1.240E-02</td>
<td>1.64%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.263E-02</td>
<td>1.240E-02</td>
<td>1.89%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.269E-02</td>
<td>1.240E-02</td>
<td>2.30%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.278E-02</td>
<td>1.240E-02</td>
<td>3.08%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.305E-02</td>
<td>1.240E-02</td>
<td>5.28%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.506E-02</td>
<td>1.240E-02</td>
<td>21.43%</td>
</tr>
</tbody>
</table>

Table 307.15: Rotation angle results for element side length 0.5 m under the inplane force.

<table>
<thead>
<tr>
<th>Poisson’s ratio</th>
<th>4NodeANDES angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.271E-02</td>
<td>1.240E-02</td>
<td>2.51%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.272E-02</td>
<td>1.240E-02</td>
<td>2.56%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.272E-02</td>
<td>1.240E-02</td>
<td>2.58%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.272E-02</td>
<td>1.240E-02</td>
<td>2.60%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.273E-02</td>
<td>1.240E-02</td>
<td>2.63%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.273E-02</td>
<td>1.240E-02</td>
<td>2.67%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.274E-02</td>
<td>1.240E-02</td>
<td>2.77%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.277E-02</td>
<td>1.240E-02</td>
<td>2.98%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.283E-02</td>
<td>1.240E-02</td>
<td>3.47%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.299E-02</td>
<td>1.240E-02</td>
<td>4.79%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.361E-02</td>
<td>1.240E-02</td>
<td>9.78%</td>
</tr>
</tbody>
</table>
Table 307.16: **Rotation angle** results for element side length 0.25 m under the inplane force.

<table>
<thead>
<tr>
<th>Poisson's ratio</th>
<th>4NodeANDES angle (unit:°)</th>
<th>Theory angle (unit:°)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.268E-02</td>
<td>1.240E-02</td>
<td>2.24%</td>
</tr>
<tr>
<td>0.05</td>
<td>1.268E-02</td>
<td>1.240E-02</td>
<td>2.27%</td>
</tr>
<tr>
<td>0.10</td>
<td>1.268E-02</td>
<td>1.240E-02</td>
<td>2.30%</td>
</tr>
<tr>
<td>0.15</td>
<td>1.269E-02</td>
<td>1.240E-02</td>
<td>2.31%</td>
</tr>
<tr>
<td>0.20</td>
<td>1.269E-02</td>
<td>1.240E-02</td>
<td>2.33%</td>
</tr>
<tr>
<td>0.25</td>
<td>1.269E-02</td>
<td>1.240E-02</td>
<td>2.35%</td>
</tr>
<tr>
<td>0.30</td>
<td>1.270E-02</td>
<td>1.240E-02</td>
<td>2.41%</td>
</tr>
<tr>
<td>0.35</td>
<td>1.271E-02</td>
<td>1.240E-02</td>
<td>2.53%</td>
</tr>
<tr>
<td>0.40</td>
<td>1.275E-02</td>
<td>1.240E-02</td>
<td>2.83%</td>
</tr>
<tr>
<td>0.45</td>
<td>1.284E-02</td>
<td>1.240E-02</td>
<td>3.58%</td>
</tr>
<tr>
<td>0.49</td>
<td>1.312E-02</td>
<td>1.240E-02</td>
<td>5.77%</td>
</tr>
</tbody>
</table>
Figure 307.34: Error scale 0% - 25%.

Figure 307.35: Error scale 0% - 100%.

Figure 307.36: 4NodeANDES cantilever beam for inplane force. Rotation angle error versus Poisson's ratio
307.6.3 Test of irregular shaped 4NodeANDES cantilever beams

Cantilever model was used as an example. Three different shapes were tested.

In the first test, the upper two nodes of each element were moved one half element size along the \( y \)-axis, while the lower two nodes were kept at the same location. The element shape was shown in Figure (307.39).

![Figure 307.37: Horizontal plane.](image)

![Figure 307.38: Vertical plane.](image)

Figure 307.39: 4NodeANDES cantilever beam for irregular Shape 1.

In the second test, the upper nodes of each element were moved 50% element size along the \( y \)-axis, while the lower nodes were moved 50% element size in the other direction along the \( y \)-axis. The element shape was shown in Figure (307.42).

In the third test, the upper two nodes of each element were moved 90% element size with different directions along the \( y \)-axis, while the lower nodes were moved 90% element size in the other direction along the \( y \)-axis. The element shape was shown in Figure (307.45).

The boundary conditions were shown in Figures (307.48), (307.51) and (307.54).

The Real-ESSI results were listed in Table (307.17).

The errors were listed in Tables (307.18) and (307.19).

The Real-ESSI model fei/DSL files for the table above are HERE.

Then, the beam was divided into small elements.

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, \( E=1E8\text{Pa} \), \( \nu = 0.0 \). Use the shear deformation coefficient \( \kappa = 1.2 \). The force direction was shown in Figure (307.55).
Theoretical displacement (bending and shear deformation):

\[
\begin{align*}
    d &= \frac{qL^4}{8EI} + \frac{qL^2}{2GA_v} \\
    &= \frac{qL^4}{8E\frac{bh^3}{12}} + \frac{qL^2}{2(1+\nu)\frac{bh}{\kappa}} \\
    &= \frac{400N/m \times 12^4m^4}{8 \times 10^8N/m^2 \times \frac{2^4}{12}m^4} + \frac{400N/m \times \frac{12^2}{2}m^2}{\frac{10^8}{2}N/m^2 \times 2m \times 2m \times \frac{5}{6}} \\
    &= 7.776 \times 10^{-3}m + 1.728 \times 10^{-4}m \\
    &= 7.9488 \times 10^{-3}m
\end{align*}
\]  

(307.4)
Figure 307.46: Horizontal plane.

Figure 307.47: Vertical plane.

Figure 307.48: 4NodeANDES cantilever beam boundary conditions for irregular Shape 1.

Figure 307.49: Horizontal plane.

Figure 307.50: Vertical plane.

Figure 307.51: 4NodeANDES cantilever beam boundary conditions for irregular Shape 2.

The Real-ESSI displacement results were listed in Table (307.20).

The errors were listed in Table (307.21).

The errors were shown in Figures (307.58), (307.61) and (307.64).

The Real-ESSI model fei/DSL files for the table above are HERE.

In this section, the beam was cut into smaller elements with element side length 0.5m and 0.25m respectively. And the element side length of the original models is 1.0m. The numerical models were shown in Figure (307.67), (307.70) and (307.73).

Number of division 1:
The Real-ESSI results for the force **perpendicular to plane (bending)** were listed in Table (307.22). The theoretical solution is 1.60E-5 m.

The Real-ESSI results for the **inplane force** were listed in Table (307.23). The theoretical solution is 1.60E-5 m.
Table 307.19: Errors for irregular shaped 4NodeANDES compared to normal shape.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Force direction</th>
<th>Normal shape</th>
<th>Shape 1</th>
<th>Shape 2</th>
<th>Shape 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4NodeANDES</td>
<td>perpendicular to plane (bending)</td>
<td>0.00%</td>
<td>0.42%</td>
<td>1.22%</td>
<td>9.12%</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>inplane force</td>
<td>0.00%</td>
<td>20.56%</td>
<td>51.87%</td>
<td>78.45%</td>
</tr>
</tbody>
</table>

Figure 307.55: Problem description for cantilever beams under uniform pressure.

Table 307.20: Results for 4NodeANDES cantilever beams of irregular shapes with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Shape</th>
<th>Force direction</th>
<th>Number of division</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape1</td>
<td>perpendicular to plane (bending)</td>
<td>7.750E-03 m</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape1</td>
<td>inplane force</td>
<td>6.822E-03 m</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape2</td>
<td>perpendicular to plane (bending)</td>
<td>7.656E-03 m</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape2</td>
<td>inplane force</td>
<td>3.875E-03 m</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape3</td>
<td>perpendicular to plane (bending)</td>
<td>6.637E-03 m</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape3</td>
<td>inplane force</td>
<td>1.555E-03 m</td>
</tr>
<tr>
<td>Theoretical solution</td>
<td></td>
<td></td>
<td>7.9488E-03 m</td>
</tr>
</tbody>
</table>

The errors were plotted in Figure (307.76).
The Real-ESSI model fei/DSL files for the table above are HERE.
Table 307.21: Errors for 4NodeANDES cantilever beams of irregular shapes with more elements.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Shape</th>
<th>Force direction</th>
<th>Number of division</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape1</td>
<td>perpendicular to plane (bending)</td>
<td>2.51%</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape1</td>
<td>inplane force</td>
<td>14.18%</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape2</td>
<td>perpendicular to plane (bending)</td>
<td>3.68%</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape2</td>
<td>inplane force</td>
<td>51.25%</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape3</td>
<td>perpendicular to plane (bending)</td>
<td>16.51%</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>shape3</td>
<td>inplane force</td>
<td>80.44%</td>
</tr>
</tbody>
</table>

Table 307.22: Results for 4NodeANDES clamped beams under the force perpendicular to plane (bending).

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element side length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 m</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>1.347E-05 m</td>
</tr>
<tr>
<td>Error</td>
<td>18.36%</td>
</tr>
</tbody>
</table>

Table 307.23: Results for 4NodeANDES clamped beams under the inplane force.

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Element side length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 m</td>
</tr>
<tr>
<td>4NodeANDES</td>
<td>1.62E-05 m</td>
</tr>
<tr>
<td>Error</td>
<td>1.70%</td>
</tr>
</tbody>
</table>
Figure 307.56: Error scale 0% - 15%.

Figure 307.57: Error scale 0% - 100%.

Figure 307.58: 4NodeANDES cantilever beam for irregular Shape 1. Displacement error versus Number of division.
Figure 307.59: Error scale 0% - 60%.

Figure 307.60: Error scale 0% - 100%.

Figure 307.61: 4NodeANDES cantilever beam for irregular **Shape 2** Displacement error versus Number of division
Figure 307.62: Error scale 0% - 80%.

Figure 307.63: Error scale 0% - 100%.

Figure 307.64: 4NodeANDES cantilever beam for irregular Shape 3. Displacement error versus Number of division.

Figure 307.65: Horizontal plane.

Figure 307.66: Vertical plane.

Figure 307.67: 4NodeANDES clamped beam with element side length 1.0m.
Figure 307.68: Horizontal plane.

Figure 307.69: Vertical plane.

Figure 307.70: 4NodeANDES clamped beam with element side length 0.5m.

Figure 307.71: Horizontal plane.

Figure 307.72: Vertical plane.

Figure 307.73: 4NodeANDES clamped beam with element side length 0.25m.
Figure 307.74: Error scale 0% - 20%.

Figure 307.75: Error scale 0% - 100%.

Figure 307.76: 4NodeANDES clamped beam for different element number. Displacement error versus number of division.
307.6.4 Verification of 4NodeANDES square plate with four edges clamped

Problem description: Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are clamped.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{E h^3}{12(1 - \nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 N \cdot m
\]  

(307.5)

The theoretical solution is

\[
d = \alpha_c \frac{qa^4}{D} = 0.00406 \times \frac{100 N/m^2 \times 20^4 m^4}{9.1575 \times 10^6 N \cdot m} = 2.2015 \times 10^{-3} m
\]  

(307.6)

where \( \alpha_c \) is a coefficient, which depends on the ratio of plate length to width. In this problem, the coefficient\(^4\) \( \alpha_c \) is 0.00406.

The 4NodeANDES were shown in Figures (307.77) - (307.82).

\[\begin{array}{c}
0 \quad 10 \quad 20m \\
10 \\
20m
\end{array}\]

Figure 307.77: 4NodeANDES edge clamped square plate with element side length 10m.

The results were listed in Table (307.24).

The errors were listed in Table (307.25).

The errors were plotted in Figure (307.85).

Figure 307.78: 4NodeANDES edge clamped square plate with element side length 5m.

Table 307.24: Results for 4NodeANDES square plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>4NodeANDES</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element side length</td>
<td>Height:1.00m</td>
<td></td>
</tr>
<tr>
<td>10m</td>
<td>2.33E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>5m</td>
<td>2.75E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>2m</td>
<td>2.58E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>1m</td>
<td>2.54E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>0.5m</td>
<td>2.53E-003 m</td>
<td>2.20E-03 m</td>
</tr>
<tr>
<td>0.25m</td>
<td>2.53E-003 m</td>
<td>2.20E-03 m</td>
</tr>
</tbody>
</table>

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 307.79: 4NodeANDES edge clamped square plate with element side length 2m.

Table 307.25: Errors for 4NodeANDES square plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>4NodeANDES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height:1.00m</td>
<td></td>
</tr>
<tr>
<td>10m</td>
<td>5.65%</td>
</tr>
<tr>
<td>5m</td>
<td>24.98%</td>
</tr>
<tr>
<td>2m</td>
<td>16.97%</td>
</tr>
<tr>
<td>1m</td>
<td>15.28%</td>
</tr>
<tr>
<td>0.5m</td>
<td>14.84%</td>
</tr>
<tr>
<td>0.25m</td>
<td>14.73%</td>
</tr>
</tbody>
</table>
Figure 307.80: 4NodeANDES edge clamped square plate with element side length 1m.

Figure 307.81: 4NodeANDES edge clamped square plate with element side length 0.5m.
Figure 307.82: 4NodeANDES edge clamped square plate with element side length 0.25m.
Figure 307.83: Error scale 0% - 25%.

Figure 307.84: Error scale 0% - 100%.

Figure 307.85: 4NodeANDES square plate with edge clamped. Displacement error versus Number of side division
307.6.5 Verification of 4NodeANDES square plate with four edges simply supported

Problem description: Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are simply supported.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 N \cdot m \tag{307.7}
\]

The theoretical solution is

\[
d = \alpha_s \frac{qa^4}{D} = 0.00126 \times \frac{100 N/m^2 \times 20^4 m^4}{9.1575 \times 10^6 N \cdot m} = 7.0936 \times 10^{-3} m \tag{307.8}
\]

where \( \alpha_s \) is a coefficient, which depends on the ratio of plate length to width. In this problem, the coefficient\(^5 \) \( \alpha_s \) is 0.00126.

The 4NodeANDES were shown in Figure (307.86) - (307.91).

![Figure 307.86: 4NodeANDES edge simply supported square plate with element side length 10m.](image)

The results were listed in Table (307.26).

The errors were listed in Table (307.27).

The errors were plotted in Figure (307.94).

---

Figure 307.87: 4NodeANDES edge simply supported square plate with element side length 5m.

Table 307.26: Results for 4NodeANDES square plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element type</th>
<th>4NodeANDES Height:1.00m</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>10m</td>
<td>1.14E-002 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>5m</td>
<td>1.03E-002 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>2m</td>
<td>9.78E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>1m</td>
<td>9.70E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>0.5m</td>
<td>9.68E-003 m</td>
<td>7.09E-03 m</td>
</tr>
<tr>
<td>0.25m</td>
<td>9.67E-003 m</td>
<td>7.09E-03 m</td>
</tr>
</tbody>
</table>

The Real-ESSI model fei/DSL files for the table above are HERE.
Figure 307.88: 4NodeANDES edge simply supported square plate with element side length 2m.

Table 307.27: Errors for 4NodeANDES square plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element side length</th>
<th>4NodeANDES</th>
</tr>
</thead>
<tbody>
<tr>
<td>10m</td>
<td>60.34%</td>
</tr>
<tr>
<td>5m</td>
<td>45.14%</td>
</tr>
<tr>
<td>2m</td>
<td>37.83%</td>
</tr>
<tr>
<td>1m</td>
<td>36.69%</td>
</tr>
<tr>
<td>0.5m</td>
<td>36.40%</td>
</tr>
<tr>
<td>0.25m</td>
<td>36.32%</td>
</tr>
</tbody>
</table>
Figure 307.89: 4NodeANDES edge simply supported square plate with element side length 1m.

Figure 307.90: 4NodeANDES edge simply supported square plate with element side length 0.5m.
Figure 307.91: 4NodeANDES edge simply supported square plate with element side length 0.25m.
Figure 307.92: Error scale 0% - 70%.

Figure 307.93: Error scale 0% - 100%.

Figure 307.94: 4NodeANDES square plate with edge simply supported' Displacement error versus Number of side division
### 307.6.6 Verification of 4NodeANDES circular plate with all edges clamped

Problem description: Diameter=20m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.3$.

The four edges are clamped.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

$$D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1-0.3^2)} = 9.1575 \times 10^6 \text{ N} \cdot \text{m}$$

(307.9)

The theoretical solution\(^6\) is

$$d = \frac{qa^4}{64D} = \frac{100N/m^2 \times 10^4 m^4}{64 \times 9.1575 \times 10^6 \text{ N} \cdot \text{m}} = 1.7106 \times 10^{-3} \text{m}$$

(307.10)

The 4NodeANDES were shown in Figures (307.95) - (307.100).

![4NodeANDES edge clamped circular plate with element side length 10m.](image)

Figure 307.95: 4NodeANDES edge clamped circular plate with element side length 10m.

The results were listed in Table (307.28).

The errors were listed in Table (307.29).

The errors were shown in Figure (307.103).

The Real-ESSI model fei/DSL files for the table above are HERE.

---

Figure 307.96: 4NodeANDES edge clamped circular plate with element side length 5m.

Figure 307.97: 4NodeANDES edge clamped circular plate with element side length 2m.
Figure 307.98: 4NodeANDES edge clamped circular plate with element side length 1m.

Figure 307.99: 4NodeANDES edge clamped circular plate with element side length 0.5m.
Figure 307.100: 4NodeANDES edge clamped circular plate with element side length 0.25m.

Table 307.28: Results for 4NodeANDES circular plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>4NodeANDES</th>
<th>Theoretical displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element side length</td>
<td>Height:1.00m</td>
<td></td>
</tr>
<tr>
<td>10m</td>
<td>1.69E-003 m</td>
<td>1.706E-03 m</td>
</tr>
<tr>
<td>5m</td>
<td>1.97E-003 m</td>
<td>1.706E-03 m</td>
</tr>
<tr>
<td>2m</td>
<td>1.97E-003 m</td>
<td>1.706E-03 m</td>
</tr>
<tr>
<td>1m</td>
<td>1.96E-003 m</td>
<td>1.706E-03 m</td>
</tr>
<tr>
<td>0.5m</td>
<td>1.96E-003 m</td>
<td>1.706E-03 m</td>
</tr>
<tr>
<td>0.25m</td>
<td>1.96E-003 m</td>
<td>1.706E-03 m</td>
</tr>
</tbody>
</table>

Table 307.29: Errors for 4NodeANDES circular plate with four edges clamped.

<table>
<thead>
<tr>
<th>Element type</th>
<th>4NodeANDES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element side length</td>
<td>Height:1.00m</td>
</tr>
<tr>
<td>10m</td>
<td>0.71%</td>
</tr>
<tr>
<td>5m</td>
<td>15.43%</td>
</tr>
<tr>
<td>2m</td>
<td>15.31%</td>
</tr>
<tr>
<td>1m</td>
<td>15.16%</td>
</tr>
<tr>
<td>0.5m</td>
<td>15.13%</td>
</tr>
<tr>
<td>0.25m</td>
<td>15.12%</td>
</tr>
</tbody>
</table>
Figure 307.101: Error scale 0% - 20%.

Figure 307.102: Error scale 0% - 100%.

Figure 307.103: 4NodeANDES circular plate with edge clamped. Displacement error versus Number of side division
307.6.7 Verification of 4NodeANDES circular plate with all edges simply supported

Problem description: Diameter=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are simply supported.

The load is the uniform normal pressure on the whole plate.

The plate flexural rigidity is

\[
D = \frac{Eh^3}{12(1-\nu^2)} = \frac{10^8 N/m^2 \times 1^3 m^3}{12 \times (1 - 0.3^2)} = 9.1575 \times 10^6 \text{ N} \cdot \text{m} \quad (307.11)
\]

The theoretical solution\(^7\) is

\[
d = \frac{(5 + \nu)qa^4}{64(1 + \nu)D} = \frac{(5 + 0.3) \times 100N/m^2 \times 10^4 m^4}{64 \times (1 + 0.3) \times 9.1575 \times 10^6 \text{ N} \cdot \text{m}} = 6.956 \times 10^{-3} m \quad (307.12)
\]

The 4NodeANDES were shown in Figure (307.104) - (307.109).

---

Figure 307.105: 4NodeANDES edge simply supported circular plate with element side length 5m.

Figure 307.106: 4NodeANDES edge simply supported circular plate with element side length 2m.
Figure 307.107: 4NodeANDES edge simply supported circular plate with element side length 1m.

Figure 307.108: 4NodeANDES edge simply supported circular plate with element side length 0.5m.
Figure 307.109: 4NodeANDES edge simply supported circular plate with element side length 0.25m.

Table 307.30: Results for 4NodeANDES cicular plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element side length</th>
<th>4NodeANDES</th>
<th>Theoretical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height: 1.00 m</td>
<td>7.50E-003 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>10 m</td>
<td>7.29E-003 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>5 m</td>
<td>7.25E-003 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>2 m</td>
<td>7.23E-003 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>1 m</td>
<td>7.22E-003 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>0.5 m</td>
<td>7.22E-003 m</td>
<td>6.956E-03 m</td>
</tr>
<tr>
<td>0.25 m</td>
<td>7.22E-003 m</td>
<td>6.956E-03 m</td>
</tr>
</tbody>
</table>
Table 307.31: Errors for 4NodeANDES circular plate with four edges simply supported.

<table>
<thead>
<tr>
<th>Element side length</th>
<th>4NodeANDES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height: 1.00 m</td>
<td></td>
</tr>
<tr>
<td>10m</td>
<td>7.75%</td>
</tr>
<tr>
<td>5m</td>
<td>4.73%</td>
</tr>
<tr>
<td>2m</td>
<td>4.15%</td>
</tr>
<tr>
<td>1m</td>
<td>3.89%</td>
</tr>
<tr>
<td>0.5m</td>
<td>3.84%</td>
</tr>
<tr>
<td>0.25m</td>
<td>3.82%</td>
</tr>
</tbody>
</table>

Figure 307.110: Error scale 0% - 8%.

Figure 307.111: Error scale 0% - 100%.

Figure 307.112: 4NodeANDES circular plate with edge simply supported. Displacement error versus Number of side division.
Chapter 308

Verification and Validation for Static and Dynamic Behavior of Special Elements (Contacts/Interfaces/Joints, Gap/Frictional, Isolators)

(In collaboration with Mr. Sumeet Kumar Sinha and Dr. Yuan Feng, and Dr. Han Yang)
308.1 Chapter Summary and Highlights

308.2 Verification of Static Penalty Contact/Interface/Joint Element Modeling and Simulation


This section presents the verification of Penalty Stiffness based Frictional Contact/Interface/Joint Element using analytical simple solutions to verify the numerical solutions obtained by the application of the developed model. The examples show the response of element for different cases. Solution sensitivity on penalty stiffness is also discussed in details for those examples.

Theoretically, the penalty stiffness should be infinite, but for numerical stability of the solution, it can go up to $10^{16}$. This is because for a double precision computer, machine epsilon $\epsilon \approx 10^{-16}$ and thus the corresponding displacement for the penalty springs can go low only till $10^{-16}$. For all the cases considered below in this cases, the convergence criteria was as $||\delta U|| \leq 10^{-12}$.

308.2.1 Static Normal Contact/Interface/Joint Verification

A Two-bar truss example is considered here to verify the normal contact/interface/joint for different normal loading conditions and different penalty stiffness $K_n$. This is an example of normal loading on a 1-D contact/interface/joint between two bars separated by an initial gap of $\delta_{in} = 0.1m$. An illustrative diagram of the problem statement is shown below.

![Illustration of two bar normal Contact/Interface/Joint problem under monotonic loading with initial gap.](image_url)

Figure 308.1: Illustration of two bar normal Contact/Interface/Joint problem under monotonic loading with initial gap.

A snapshot of the code for the contact/interface/joint element is shown below. The Real-ESSI model fei/DSL files for this example can be downloaded [HERE](#).

```plaintext
1 add element #3 type FrictionalPenaltyContact with nodes (2,3)
2 normal_stiffness = 1e10*N/m
3 tangential_stiffness = 1e10*Pa*m
4 normal_damping = 0*kN/m*s
5 tangential_damping = 0*kN/m*s
6 friction_ratio = 0.3
7 contact_plane_vector = (1,0,0);
```
308.2.1.1 **Case 1: Monotonic Loading with initial gap** $\delta_{in} = 0.1m$

In this case a force of 0.3N is applied to Node 2. From Figure 308.2, the solution converges to the analytical result for $K_n = 100 N/m$ i.e 100 times the stiffness of bar element. Please note that, the for penalty stiffness $> 10^{15}$, the convergence fails (when the bars contact/interface/joint mode changes), as the global stiffness matrix becomes *ill conditions*. Thus, the penalty stiffness cannot be too large.

308.2.1.2 **Case 2: Monotonic Loading with no initial gap** $\delta_{in} = 0m$

In this case a force of 0.3N is applied to Node 2. From Figure 308.3, the solution again converges to the analytical result for $K_n = 100 N/m$ i.e 100 times the stiffness of bar element.

308.2.1.3 **Case 3: Cyclic Loading with initial gap** $\delta_{in} = 0.1m$

For cyclic loading cases considered below, the loading force $F_n$ applied is shown in Figure 308.4. From Figure 308.5, the solution again converges to the analytical result for $K_n = 100 N/m$ i.e 100 times the stiffness of bar element.

308.2.1.4 **Case 4: Cyclic Loading with no initial gap** $\delta_{in} = 0m$

The same cyclic load shown in Figure 308.4 is again applied for this case. From Figure ??, the solution again converges to the analytical result for $K_n = 100 N/m$ i.e 100 times the stiffness of bar element.
Figure 308.2: Displacements of Node 2 and Node 3 with change in normal penalty stiffness for $\delta_{in} = 0.1m$
Figure 308.3: Displacements of Node 2 and Node 3 with change in normal penalty stiffness for $\delta_{in} = 0m$
Figure 308.4: Cyclic normal load applied on two bar contact/interface/joint problem.
Figure 308.5: Displacements of Node 2 and Node 3 with change in normal penalty stiffness for $\delta_{in} = 0 \text{m}$
Figure 308.6: Displacements of Node 2 and Node 3 with change in normal penalty stiffness for $\delta_{in} = 0m$
308.2.2 Static Frictional Tangential Contact/Interface/Joint Verification

A simple 3-D truss example with Normal confinement in z-direction of $F_N = 0.5N$, friction coefficient $\mu = 0.2$ and shear loading of magnitude $F_s = 0.5N$ is considered to verify the tangential behaviour of contact/interface/joint element. Different cases as discussed below are considered.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure308.7}
\caption{Illustration of 3-D three bar contact/interface/joint problem.}
\end{figure}

A snapshot of the properties of contact/interface/joint element are shown below. The Real-ESSI model fei/DSL files for this example can be downloaded [HERE](#).

```plaintext
1  add element #4 type FrictionalPenaltyContact with nodes (1,2)
2  normal_stiffness = 1e10*N/m
3  tangential_stiffness = 1e10*Pa*m
4  normal_damping = 0*kN/m*s
5  tangential_damping = 0*kN/m*s
6  friction_ratio = 0.2
7  contact_plane_vector = (0,0,1);
```
308.2.2.1 Case 1: Verification of the yield surface for different loading angles with fixed normal confinement.

A Shear force of magnitude of \( F_s = 0.5N \) was applied in 20 steps in different loading directions. The response of the contact/interface/joint element and the displacement of node 2 is shown in Figure 308.8 and Figure 308.9 respectively. It can be observed that the contact/interface/joint element slips at magnitude of force \( ||F|| - (F_n = 0.5) \ast (\mu = 0.2) - > 0.1N \) for all loading angles.

In Figure 308.9, it can be observed that for the first 4 steps, there is no (zero) displacement for node 2 because of the stick case. When the load exceeds 0.1N, slip occurs and node 2 starts to undergo deformation.
Figure 308.8: Response of contact/interface/joint element for different loading angles for confinement of $0.5\,N$ and coefficient of friction as 0.2.
Figure 308.9: Displacement of Node No. 2 in x and y direction for different loading angles for confinement of 0.5 N and coefficient of friction as 0.2.
308.3 Verification of Static and Dynamic Contact/interface/Joint Element Modeling and Simulation

Solution verification of the contact/interface/joint element formulation and its implementation is presented in what follows. Analytical simple solutions for the frictional contact/interface/joint element are used to verify numerical solutions obtained by application of the developed model. The examples provided show the response of the contact element in several situations. Initially, the element is tested by connecting two nodes that each have 3dof, subsequently, the element is even implemented to simulate a contact/interface/joint between two nodes that each have 7dof and two nodes with different dofs: 3dof for the first one and 7dof for the second one.

The parameters used for the contact/interface/joint element are listed in Table 308.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_N$ [kN/m]</td>
<td>10420</td>
</tr>
<tr>
<td>$v_{max}$ [m]</td>
<td>0.001</td>
</tr>
<tr>
<td>$K_T$ [kN/m]</td>
<td>$1e7$</td>
</tr>
<tr>
<td>$\mu$ [-]</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 308.1: Contact/interface/joint element parameters.

Figure 308.10: Input signal: time history of displacement, acceleration and velocity.
308.3.1 Truss Examples

The first example (Figure 308.11) represents five nodes: 1, 2, 3, 4, 5. Nodes 2, 3, 4, 5 are connected by three truss elements and a contact/interface/joint element links node 1 and 2. All degrees of freedom of nodes 1, 3, 4, 5 are fixed, whereas a sine wave displacement time-history (Figure 308.10) is applied to node 2 along x direction and the normal force acting within the contact/interface/joint element is recorded. The results, represented in Figure 308.12, show the normal response of this new contact/interface/joint element. As the timestep decreases, the force-displacement curves tends to be similar to the one represented by (??).

Figure 308.11: System composed of one contact/interface/joint element and three truss elements. A sine wave displacement time-history is applied to node (2).

Figure 308.12: Normal force vs normal-relative displacement in contact/interface/joint element.
The second example (Figure 308.13) shows the tangential response of this new contact/inter-
face/joint element. The geometry of the problem is the same as Figure 308.11, but an axial force
\( p = 1140 \text{ kN} \), constant in time, and a sine wave time-history displacement are applied to node 2. The
results, represented in Figure 308.14, show that the response is not dependent on the timestep used for
the analysis. Due to the elastic-perfectly-plastic behavior, associated with Mohr-Coulomb yield criteria,
the maximum shear force \( t_{max} \) that the contact/interchange/joint element can sustain is 684 kN , equal
to \( t_{max} = \mu \cdot p = 0.6 \cdot 1140 \text{kN} \).

![Figure 308.13: Same system used in Figure 308.11. A normal force constant in time (\( F = 1140 \text{kN} \)) and a sine wave displacement time history are applied to node (2).](image1)

![Figure 308.14: \( p = 1140 \text{kN} \). Transversal force in the contact/interchange/joint element vs transversal-relative displacement](image2)
308.3.2 Single Brick Element Examples

The contact/interface/joint element is now used to connect each node (5, 6, 7, 8) of the bottom face of an eight node hexahedral element (hex8) to the corresponding node attached to the ground (1, 2, 3, 4).

A linear elastic constitutive model is used for the eight node brick and the parameters are listed in Table 308.2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$ [kPa]</td>
<td>1.5e10</td>
</tr>
<tr>
<td>$\nu$ [-]</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 308.2: Brick Element Parameters

In the first example (Figure 308.15) the time-history displacement, shown in Figure 308.10, is applied in the vertical direction to each node of the top surface of the brick element (9, 10, 11, 12) and the time-history normal force induced in each contact element is represented in Figure 308.16. It is worth noting that the normal force is positive if the displacement of the top surface is downward, whereas it is zero if the detachment occurs caused by the upward movement.

Through the second example, shown in Figure 308.17, the transversal response of the contact/interface/joint elements is highlighted. The vertical normal force ($F_v = 50kN$), constant in time, and a horizontal time-history displacement are applied to each node of the top surface and the transversal response is shown in Figure 308.18. As stated in section 308.3.1, the maximum shear force that the contact element can sustain is 30 kN, equal to $t_{max} = \mu \cdot p = 0.6 \cdot 50kN$.

The third example is focused on the transversal response of the contact/interface/joint element under variable normal forces. In fact, a sine wave time-history horizontal displacement and a vertical force are applied to each node of the top surface of the brick. The normal force is variable in time according to the factor ($Fact(t)$) shown in Figure 308.20 and the vertical force is computed as $F_v(t) = Fact(t) \cdot F_{v,max}$, and $F_{v,max}$ equal to 50 kN. The response of the contact element, shown in Figure 308.21, is independent of the timestep used for the analysis emphasizing the correct numerical implementation.

308.3.3 Double Brick Element Examples

Few other examples are produced taking in consideration two brick elements. The constitutive model used for these two brick elements is linear elastic with the same parameters listed in Table 308.2. Vertical and horizontal time-history displacement are applied to the top surface, shown in Figure 308.22 and Figure 308.23, and variable vertical forces are considered in the example represented in Figure 308.24. The results are the same shown in Figure 308.16, Figure 308.18 and Figure 308.19.
Figure 308.15: Eight-node brick element over four contact/interface/joint elements. A sine wave time-history vertical displacement applied to each node of the top surface.

Figure 308.16: Normal force vs time in each contact element.
Figure 308.17: Eight-node brick element over four contact/interface/joint elements. $F_v = 50$ kN and a sine wave time-history horizontal displacement applied to each node of the top surface.

Figure 308.18: Transversal force vs transversal relative-displacement in the contact element with normal force $F$ equal to 50 kN.
Figure 308.19: Eight-node brick element over four contact/interface/joint elements. Variable vertical force \( F_v(t) \) and a sine wave time-history horizontal displacement applied to each node of the top surface.

Figure 308.20: Time-history of the normal force factor \( F_{act}(t) \).
Figure 308.21: Variable normal force. Transversal force vs transversal relative-displacement in each contact element.

Figure 308.22: Two eight-node brick elements connected by four contact/interface/joint elements. Vertical time-history displacement applied to the nodes of the top surface.
Figure 308.23: Two eight-node brick elements connected by four contact/interface/joint elements. Vertical force, equal to 50 kN and constant in time, and horizontal time-history displacement applied to the nodes of the top surface.

Figure 308.24: Two eight-node brick elements connected by four contact/interface/joint elements. Vertical force, variable in time, and horizontal time-history displacement applied to the nodes of the top surface.
308.4 Verification of Static and Dynamic Coupled (Saturated) Contact/Interface/Joint Element Modeling and Simulation

308.4.1 Dry u-p-U Contact/Interface/Joint to the Ground

A single brick $u$-$p$-$U$ finite element is used to model an oedometric compression shown in Figure 308.25. Horizontal displacements and pore pressure are fixed in each node in order to guarantee the one-dimensional and dry conditions. As the ground is modeled as an undeformable and impermeable layer (1, 2, 3, 4), vertical soil and fluid displacements are fixed. The time-history displacement, shown in Figure 308.10, is applied in vertical direction to each node of the top surface (9, 10, 11, 12).

The time-history normal force induced in each contact/interface/joint element is represented in Figure 308.16 and compared with the one obtained with the dry brick element with $u$ formulation, shown in Figure 308.15. It is worth noting that the normal force patterns are perfectly overlapped: this is due to the fact that excess pore pressure is fixed to zero and the oedometric stiffness are the same in the two cases.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young’s Modulus</td>
<td>$E$ [kPa]</td>
<td>$1.5 \cdot 10^{10}$</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>$\nu$ [-]</td>
<td>0.0</td>
</tr>
<tr>
<td>Solid particle bulk modulus</td>
<td>$K_s$ [kPa]</td>
<td>$3.6 \cdot 10^{7}$</td>
</tr>
<tr>
<td>Fluid bulk modulus</td>
<td>$K_f$ [kPa]</td>
<td>$2.17 \cdot 10^{6}$</td>
</tr>
<tr>
<td>Solid density</td>
<td>$\rho_s$ [Mg/m$^3$]</td>
<td>2.7</td>
</tr>
<tr>
<td>Fluid density</td>
<td>$\rho_f$ [Mg/m$^3$]</td>
<td>1.0</td>
</tr>
<tr>
<td>Porosity</td>
<td>$n$ [-]</td>
<td>$1.0 \cdot 10^{-8}$</td>
</tr>
<tr>
<td>Darcy permeability</td>
<td>$K$ [m/s]</td>
<td>$1.0 \cdot 10^{3}$</td>
</tr>
</tbody>
</table>

Table 308.3: Soil parameters.
Figure 308.25: Single eight-node brick element. The nodes of the bottom surface are connected to the ground floor through contact/interface/joint elements. Vertical time-history displacement applied to the nodes of the top surface.

Figure 308.26: Normal force vs time in each contact/interface/joint element. $u$-formulation represents the results shown in Figure 308.16 whereas up$U$-formulation represents the results obtained through the model shown in Figure 308.25.
308.4.2 u-p-U Contact/Interface/Joint to the ground

A column of four brick \textit{u-p-U} finite elements is used to model the horizontal layer. The height of the soil column is 1 m and the height of each element has dimensions $1\text{m} \times 1\text{m} \times 1\text{m}$, illustrated in Figure 308.27.

The following boundary conditions are applied to the model. As the bottom of the column is modeled as an undeformable and impermeable layer, both the solid and fluid displacements are fixed. The pore pressure is kept constant as zero at the top surface of the soil column because of the perfectly drained condition. In order to simulate the 1D compression problem, all the lateral movement of the solid and fluid phase are constrained so that the vertical displacement is the only non-zero displacement. A vertical time-history displacement, shown in Figure 308.10, is applied to the solid \textit{dof} of the top surface nodes. In order to simulate the one dimensional compression, all the degrees of freedom at the same level are connected in a masterslave fashion.

The nodes of the bottom surface (5, 6, 7, 8) are connected to the ground (1, 2, 3, 4) through contact/interface/joint elements. Under the hypothesis of laminar flow, no cavitation and one-dimensional type of problem, the water has to fill up all the voids generated into the media while the displacement filed is acting on the top surface. This means that if the soil and the ground are separated because of the detachment of the contact/interface/joint element, the water has to fill the gap. Even the pore pressure at both sides of the gap has to assume the same value. Such boundary conditions can be introduced by adding masterslave between each node of the bottom surface (5, 6, 7, 8) and the corresponding one belonging to the ground (1, 2, 3, 4).

In this paragraph a parametric study is performed in order to examine the performance of this \textit{u-p-U} contact/interface/joint element. The analyses consider several configuration of permeability and soil stiffness. In fact, three values of elastic modulus ($E$) are taken into consideration referring to a Stiff, Medium and Soft soil (called respectively StS, MS and SoS) and three values of the darcy permeability are similarly defined. All soil and contact/interface/joint parameters are listed respectively in Table 308.4 and Table 308.1.

This kind of excitation (a sine wave) applied at the top of the model is clearly composed of waves of all kinds of frequency, expetially at the initial strong change in acceleration and velocity for $t = 0\text{s}$. Due to this fact, a fairly dense mesh of 100 \textit{u-p-U} brick finite elements of dimensions $1\text{m} \times 1\text{m} \times 1\text{cm}$ was chosen. Therefore, the time step $\Delta t$ needs to be limited to $\Delta t = \Delta h/v$, where $v$ is the highest wave velocity. In our case, the temporal integration involves 4000 steps of , which allows a maximum wave velocity of . The propagation velocity can be calculated by the following equation given by de Boer et
al. (1993) and is equal to

$$v = \sqrt{\frac{n^2(1-\nu)E}{(1+\nu)(1-2\nu)[n^2(1-n)\rho_s + (1-n)^2n\rho_f]}}$$

(308.1)

The artificial oscillation are diminished by introducing some numerical damping into the analysis through $\alpha = 0.6$ and $\beta = 0.3025$ for the Newmark integrator.

Figure 308.27: Four eight-node brick elements. The nodes of the bottom surface are connected to the ground floor through contact/interface/joint elements. Vertical time-history displacement applied to the nodes of the top surface.

The specific permeability, $k$ and the time needed for completion of the 1D consolidation process, $t$, can be estimated using the darcy permeability $K$ through (308.2) and (308.3) respectively, where $\rho_w$ is the mass density of the fluid (water), $g$ is the acceleration of gravity equal to 9.81 m/s$^2$, $H$ is the thickness of the soil layer and $E_{oed}$ is the one dimensional soil stiffness:

$$k = \frac{K}{\rho_w \cdot g}$$

(308.2)

$$t = \frac{H^2}{c_v} = \frac{H^2 \cdot \rho_w \cdot g}{K \cdot E_{oed}}$$

(308.3)

The time $t$ is computed and listed in Table 308.4.2 for each soil type condition. As can be seen, for the stiff soil (StS), the consolidation time is much lower than the loading acting on the top surface, therefore no excess pore pressure is developed into the soil and drained response is occurred.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STIFF SOIL - (StS)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Young’s Modulus</td>
<td>$E$ [kPa]</td>
<td>$1.0 \cdot 10^{10}$</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>$\nu$ [-]</td>
<td>0.0</td>
</tr>
<tr>
<td>Solid particle bulk modulus</td>
<td>$K_s$ [kPa]</td>
<td>$3.6 \cdot 10^7$</td>
</tr>
<tr>
<td>Fluid bulk modulus</td>
<td>$K_f$ [kPa]</td>
<td>$2.17 \cdot 10^6$</td>
</tr>
<tr>
<td>Solid density</td>
<td>$\rho_s$ [Mg/m$^3$]</td>
<td>2.7</td>
</tr>
<tr>
<td>Fluid density</td>
<td>$\rho_f$ [Mg/m$^3$]</td>
<td>1.0</td>
</tr>
<tr>
<td>Porosity</td>
<td>$n$ [-]</td>
<td>0.46</td>
</tr>
<tr>
<td>Darcy permeability</td>
<td>$K$ [m/s]</td>
<td>$1.0 \cdot 10^{-3}$</td>
</tr>
<tr>
<td><strong>MEDIUM SOIL - (MS)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Young’s Modulus</td>
<td>$E$ [kPa]</td>
<td>$1.0 \cdot 10^7$</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>$\nu$ [-]</td>
<td>0.0</td>
</tr>
<tr>
<td>Solid particle bulk modulus</td>
<td>$K_s$ [kPa]</td>
<td>$3.6 \cdot 10^7$</td>
</tr>
<tr>
<td>Fluid bulk modulus</td>
<td>$K_f$ [kPa]</td>
<td>$2.17 \cdot 10^6$</td>
</tr>
<tr>
<td>Solid density</td>
<td>$\rho_s$ [Mg/m$^3$]</td>
<td>2.7</td>
</tr>
<tr>
<td>Fluid density</td>
<td>$\rho_f$ [Mg/m$^3$]</td>
<td>1.0</td>
</tr>
<tr>
<td>Porosity</td>
<td>$n$ [-]</td>
<td>0.46</td>
</tr>
<tr>
<td>Darcy permeability</td>
<td>$K$ [m/s]</td>
<td>$1.0 \cdot 10^{-3}$</td>
</tr>
<tr>
<td><strong>SOFT SOIL - (SoS)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Young’s Modulus</td>
<td>$E$ [kPa]</td>
<td>$1.0 \cdot 10^4$</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>$\nu$ [-]</td>
<td>0.0</td>
</tr>
<tr>
<td>Solid particle bulk modulus</td>
<td>$K_s$ [kPa]</td>
<td>$3.6 \cdot 10^7$</td>
</tr>
<tr>
<td>Fluid bulk modulus</td>
<td>$K_f$ [kPa]</td>
<td>$2.17 \cdot 10^6$</td>
</tr>
<tr>
<td>Solid density</td>
<td>$\rho_s$ [Mg/m$^3$]</td>
<td>2.7</td>
</tr>
<tr>
<td>Fluid density</td>
<td>$\rho_f$ [Mg/m$^3$]</td>
<td>1.0</td>
</tr>
<tr>
<td>Porosity</td>
<td>$n$ [-]</td>
<td>0.46</td>
</tr>
<tr>
<td>Darcy permeability</td>
<td>$K$ [m/s]</td>
<td>$1.0 \cdot 10^{-3}$</td>
</tr>
</tbody>
</table>

Table 308.4: Stiff, Medium and Soft soil parameters (StS, MS and SoS).
Table 308.5: The time needed for completion of the 1D consolidation process, $t$, estimated through the Darcy permeability $K$ for each soil type condition.

<table>
<thead>
<tr>
<th>Soil type</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K [$m/s$]</td>
<td>1.0 · 10$^{-3}$</td>
<td>1.0 · 10$^{-5}$</td>
</tr>
<tr>
<td>Stiff soil (StS)</td>
<td>$t$ [s]</td>
<td>1.0 · 10$^{-6}$</td>
</tr>
<tr>
<td>Medium-Stiff soil (MS)</td>
<td>$t$ [s]</td>
<td>1.0 · 10$^{-3}$</td>
</tr>
<tr>
<td>Soft soil (SoS)</td>
<td>$t$ [s]</td>
<td>1.0 · 10$^{0}$</td>
</tr>
</tbody>
</table>
308.5 Verification of Static, Isolator Element Modeling and Simulation
Chapter 309

Verification and Validation for Coupled, Porous Solid – Pore Fluid Problems

(In collaboration with Prof. Zhao Cheng, Dr. Panagiota Tasiopoulou, Ms. Fatemah Behbehani, Dr. Han Yang and Mr. Yusheng Yang)
309.1 Chapter Summary and Highlights

309.2 Introduction

Presented here are verification examples for u-p-U formulation. Examples include:

1. Drilling of a borehole
2. The case of a spherical cavity
3. Consolidation of a soil layer
4. Line injection of a fluid in a reservoir
5. Shock wave propagation
8. One dimensional shock wave propagation with step loading at the surface by de Boer et al. (1993)

309.3 Drilling of a well

309.3.1 The Problem

Let us consider an infinite half space domain composed of an isotropic, homogeneous and saturated thermoporoelastic material. At its reference state, it is assumed that the temperature, fluid pressure and stress fields are uniform and equal respectively, to \( T_0, p_0 \) and \( \sigma^0 = \sigma^0 1 \) (with \( \sigma^0 < 0 \)). At time = 0, an infinite cylinder of radius \( r_0 \) is instantaneously drilled parallel to the vertical axis \( Oz \). It is filled with a fluid of the same nature as that saturating the porous medium but at a different pressure and temperature at the values of \( p_1 \) and \( T_1 \) respectively. The interface \( r = r_0 \) between the well and the porous medium is assumed to be in thermodynamic equilibrium.
In cylindrical coordinates \((r, \theta, z)\), the boundary conditions can be summarized as follows (see Fig.309.1):

\[
\begin{align*}
  t \leq 0 & \rightarrow \quad \sigma^0 = \sigma^0 1 \quad p(r) = p_0 \quad T(r) = T_0 \\
  t > 0 & \rightarrow \quad \sigma_{rr}(r_0) = -p_1 \quad \sigma_{r\theta}(r_0) = \sigma_{rz}(r_0) = 0 \\
  \sigma_{rr}(r \to \infty) & \to \sigma^0 \quad \sigma_{r\theta}(r \to \infty) = \sigma_{rz}(r \to \infty) \to 0 \\
  p(r_0, t) & = p_1 \quad p(r \to \infty) \to p_0 \\
  T(r_0) & = T_1 \quad T(r \to \infty) \to T_0
\end{align*}
\]  

(309.1)

\[
\begin{align*}
  \sigma_{rr}(r \to \infty) & \to \sigma^0 1 \quad \sigma_{r\theta}(r \to \infty) = \sigma_{rz}(r \to \infty) \to 0 \\
  p(r_0, t) & = p_1 \quad p(r \to \infty) \to p_0 \\
  T(r_0) & = T_1 \quad T(r \to \infty) \to T_0
\end{align*}
\]  

(309.2)

Figure 309.1: Boundary Conditions for Drilling of a Borehole

### 309.3.2 Analytical Solution

Since the well is assumed to be infinite long in its vertical axis \(Oz\), the analysis is performed under plane strain hypothesis\((\epsilon_{zz} = 0)\). Therefore,

\[
\begin{align*}
  \xi &= \xi^r(r)e_r \quad p = p(r) \quad T = T(r)
\end{align*}
\]  

(309.3)

in which \(\xi^r\) is the radial displacement. In cylindrical coordinates, Eqn. 309.48 yields

\[
\begin{align*}
  \epsilon_{rr} = \frac{\partial \xi^r}{\partial r} \quad \epsilon_{\theta\theta} = \frac{\xi^r}{r} \quad \text{other} \quad \epsilon_{ij} = 0
\end{align*}
\]  

(309.4)

Based on the constitutive equations from Coussy (1995), it follows that

\[
\sigma_{rr} = \sigma^0 + \lambda_0 \left( \frac{\partial \xi^r}{\partial r} + \frac{\xi^r}{r} \right) + 2\mu \frac{\partial \xi^r}{\partial r} - b(p - p_0) - 3\alpha K_0(T - T_0)
\]  

(309.5)
\[ \sigma_{\theta\theta} = \sigma^0 + \lambda_0 \left( \frac{\partial \xi^r}{\partial r} + \frac{\xi^r}{r} \right) + 2\mu \frac{\xi^r}{r} - b(p - p_0) - 3\alpha K_0(T - T_0) \]  

(309.6)

\[ \sigma_{zz} = \sigma^0 + \lambda_0 \left( \frac{\partial \xi^r}{\partial r} + \frac{\xi^r}{r} \right) - b(p - p_0) - 3\alpha K_0(T - T_0) \]  

(309.7)

\[ \text{other } \sigma_{ij} = 0 \]  

(309.8)

Finally combined with the Eqns. 309.5-309.8, it yields the near field or long-term solution (Coussy, 1995)

\[ \xi^r = \frac{\sigma^0 + \rho_1 r_0^2}{2\mu} \frac{r}{r} + r_0 \left[ \frac{b(p_1 - p_0) + 3\alpha_0 K_0(T_1 - T_0)}{2(\lambda_0 + 2\mu)} \right] \left( \frac{r}{r_0} - \frac{r_0}{r} \right) \]  

(309.9)

\[ \sigma_{rr} = -\rho_1 \frac{r_0^2}{r^2} - \left( \sigma^0 - \left[ \frac{b(p_1 - p_0) + 3\alpha_0 K_0(T_1 - T_0)}{\lambda_0 + 2\mu} \right] \right) \left( 1 - \frac{r_0^2}{r^2} \right) \]  

(309.10)

\[ \sigma_{\theta\theta} = (2\sigma_0 + \rho_1) \frac{r_0^2}{r^2} - \frac{b(p_1 - p_0) + 3\alpha_0 K_0(T_1 - T_0)}{\lambda_0 + 2\mu} \left( 1 + \frac{r_0^2}{r^2} \right) \]  

(309.11)

\[ \sigma_{zz} = \sigma^0 - \frac{2\mu}{\lambda_0 + 2\mu} \left[ b(p_1 - p_0) + 3\alpha_0 K_0(T_1 - T_0) \right] \]  

(309.12)

And the diffusion process can be achieved if that the time are large enough with respect to the characteristics diffusion time relative to point \( r \). When the boundary conditions for \( r = r_0 \) in fluid pressure and temperature which are \( p = p_1 \) and \( T = T_1 \) apply for the whole model, the following equations correspond to the undrained solution of the instantaneous drilling of a borehole in an infinite elastic medium.

\[ \xi^r = \frac{\sigma^0 + \rho_1 r_0^2}{2\mu} \frac{r}{r} \quad \sigma_{rr} = \sigma^0 - (\sigma^0 + \rho_1) \frac{r_0^2}{r^2} \]  

\[ \sigma_{\theta\theta} = \sigma^0 + (\sigma^0 + \rho_1) \frac{r_0^2}{r^2} \quad \sigma_{zz} = \sigma_0 \]  

(309.13)

### 309.3.3 Discussion of the Results

As the problem is Axisymmetric, we construct the model as a quarter of a donut. The inside diameter of the donut is 10 cm and the outside diameter is 1 m. To accommodate both the plain strain hypothesis and the geometry of the element for finite element, the thickness of the model is chosen to be 5 cm. The final mesh is generated as Fig.309.2. And the boundary conditions is as follows: As a consequence of plain strain problem, all the movements for solid and fluid in vertical direction \( O_z \) are suppressed; the solid and fluid displacement for the nodes along the X axis and Y axis are fixed in Y and X direction respectively for the reason of axisymmetry; the nodes along the outside perimeter are fixed in the solid and fluid displacement with the assumption of infinite medium. The pressure is translated into nodal forces.
Parameter | Symbol | Value | Units
--- | --- | --- | ---
Poisson Ratio | $\nu$ | 0.2 | -
Young’s Modulus | $E$ | 1.2E+6 | $kN/m^2$
Solid Bulk Modulus | $K_s$ | 3.6E+7 | $kN/m^2$
Fluid Bulk Modulus | $K_f$ | 1.0E+17 | $kN/m^2$
Solid Density | $\rho_s$ | 2.7 | $ton/m^3$
Fluid Density | $\rho_f$ | 1.0 | $ton/m^3$
Porosity | $n$ | 0.4 | -

Table 309.1: Material Properties used to study borehole problem

and applied on the nodes along the inside perimeter. For simplicity, the hydrostatic stress $\sigma_0$ is equal to zero and with the assumption of thermodynamic equilibrium through the process, the temperature factor can be neglected. Also the initial fluid pressure $p_0$ is set to be 0 kPa. The analytical solution is studied below using the following set of parameters shown in Table 309.1.

![Figure 309.2: The mesh generation for the study of borehole problem](image)

I) PLAN VIEW  II) ISOMETRIC VIEW

In the analysis, ten loading cases for final fluid pressure from 10 kPa to 100 kPa are studied. And by manipulating the permeability, it is possible to investigate both the drained behavior and undrained behavior. For the drained behavior, we choose the permeability as $k = 3.6 \times 10^{-4} m/s$, which is a typical value for sand, the comparison between the close solution and experimental result is shown in Fig.309.3. From the results, we can see that along the inside perimeter, the close solution and experimental result
provide very good agreement to each other. But as the increase of the radius, we can see the analytical solution is getting more and more distant from the experimental results. In another word, the analytical solution can be interpreted as that with the increase from the loading surface, the radial displacement is larger. This is unreasonable in the point of view in soil mechanics. While the experimental result show the effect that with the increase of the radial distance, the radial displacement is decreasing.

![Graph showing comparison of radial solid displacement between analytical solution and experimental result for drained behavior.](image)

**Figure 309.3:** The comparison of radial solid displacement between analytical solution and experimental result for drained behavior

For the undrained behavior, the permeability of \( k = 3.6 \times 10^{-8} \text{m/s} \) is selected as a representative value for typical clayey soil. The comparison between the close solution and experimental result is provided as well. From the Fig. 309.4 we can see that, the analytical solution is linearly away from the experimental result by a ratio of approximately 1.6. It should also be noticed that the close solution of the drained and undrained behavior for the nodes along the inside perimeter are exactly the same, which is contradictory to the definition of drained and undrained behavior. For the drained behavior, as the water easily dissipate from the soil body, the problem can be treated with the knowledge of continuum mechanics using the parameters of the solid skeleton. While for the undrained behavior, with the involvement of the pore water, the elastic parameters for the mixture should be different, so the response will not be the same as well. As a result of this, the experimental results give a more reasonable conclusion.
As for the drained behavior, the fluid totally flows out of the soil body and all excessive pore pressure dissipates, there is small coupling between the solid and fluid phase. We can use the continuum mechanics to treat this problem. Here introduces a problem of an infinite cylindrical tube, with the inner radius \( R_1 \) and outer radius \( R_0 \), subjected to an internal pressure \( P_1 \) and an external pressure \( P_2 \). The displacement field as follows (S. Timoshenko and D.H. Young, 1940):

\[
\xi^r = \frac{R_1^2 P_1}{2(R_0^2 - R_1^2)} \left( \frac{r}{\lambda + \mu} + \frac{R_0^2}{\mu r} \right) \quad (309.14)
\]

With \( P_0 = 0 \) and take the limit of \( R_0 \to \infty \), we can obtain the following equation:

\[
\xi^r = \frac{P_1 R_1^2}{2\mu} \frac{r}{R_0^2} \quad (309.15)
\]

which is identical to Eq.309.10. Also to minimize the effect of infinite boundary, we introduce the result from another model which is exactly the same as the previous one besides the expansion of the outer radius to 30m. At the final fluid pressure of 50 kPa, the results are shown in Fig.309.5. From the plot we can make a conclusion that the undrained analytical solution from Coussy (1995) is actually the drained solution and the undrained solution still needs to be investigated.
309.4 The Case of a Spherical Cavity

309.4.1 The Problem

Considering a medium composed of an isotropic, homogeneous, saturated thermoporoelastic material. In its initial state, it is assumed that the temperature, fluid pressure and stress fields are uniform and equal respectively, to $T_0$, $p_0$ and $\sigma^0 = \sigma^0 T$ (with $\sigma^0 < 0$). At time $t=0$, a spherical cavity of radius $r_0$ is immediately drilled and filled with the same saturating fluid in the medium. For $t > 0$, the temperature and the pressure of the fluid are kept constant with the value of $T_1$ and $p_1$ respectively. The interface $r = r_0$ between the well and the porous medium is assumed to be in the thermodynamic equilibrium.

In spherical coordinates $(r, \theta, \varphi)$, the boundary conditions can be summarized as follows:
\[ t \leq 0 \rightarrow \sigma^0 = \sigma_0^0 I \quad p(r) = p_0 \quad T(r) = T_0 \]  \hspace{1cm} (309.16)

\[ t > 0 \rightarrow \sigma_{rr}(r_0) = -p_1 \quad \sigma_{r\theta}(r_0) = \sigma_{r\phi}(r_0) = 0 \]

\[ \sigma_{rr}(r \to \infty) \to \sigma_0^0 \quad \sigma_{r\theta}(r \to \infty) = \sigma_{r\phi}(r \to \infty) \to 0 \]

\[ p(r_0, t) = p_1 \quad p(r \to \infty) \to p_0 \]

\[ T(r_0) = T_1 \quad T(r \to \infty) \to T_0 \]  \hspace{1cm} (309.17)

Strictly speaking, the expressions for \( r \to \infty \) are not boundary conditions. They are complementary conditions to be satisfied by the solution. It is used to model that at the point far from the disturbed area, the state of the medium are held as its initial state.

### 309.4.2 Analytical Solution

This is a problem of spherical symmetry. The radial displacement is the only non-zero displacement and all the fields are \( r \) and \( t \) dependent. Therefore,

\[ \xi = \xi^r(r)e_r \quad p = p(r) \quad T = T(r) \]  \hspace{1cm} (309.18)

in which \( \xi^r \) is the radial displacement. In spherical coordinates, Eqn.309.18 yields

\[ \epsilon_{rr} = \frac{\partial \xi^r}{\partial r} \quad \epsilon_{\theta\theta} = \frac{\xi^r}{r} \quad \text{other} \quad \epsilon_{ij} = 0 \]  \hspace{1cm} (309.19)

Based on the constitutive equations from Coussy (1995), it follows that

\[ \sigma_{rr} = \sigma^0 + \lambda_0 \left( \frac{\partial \xi^r}{\partial r} + \frac{\xi^r}{r} \right) + 2\mu \frac{\partial \xi^r}{\partial r} - b(p - p_0) - 3\alpha K_0(T - T_0) \]  \hspace{1cm} (309.20)

\[ \sigma_{\theta\theta} = \sigma_{r\phi} = \sigma^0 + \lambda_0 \left( \frac{\partial \xi^r}{\partial r} + \frac{\xi^r}{r} \right) + 2\mu \xi^r + b(p - p_0) - 3\alpha K_0(T - T_0) \]  \hspace{1cm} (309.21)

\[ \text{other} \quad \sigma_{ij} = 0 \]  \hspace{1cm} (309.22)

Finally combined with the Eqns. 309.19-309.22, it yields the near field or long-term solution (Coussy, 1995)

\[ \xi^r = \frac{\sigma^0 + p_1 r_0^3}{4\mu} \frac{1}{r^2} + \frac{r_0 b(p_1 - p_0) + 3\alpha K_0(T_1 - T_0)}{2(\lambda_0 + 2\mu)} \left( 1 - \frac{r_0^2}{r^2} \right) \]  \hspace{1cm} (309.23)

\[ \sigma_{rr} = -p_1 \frac{r_0^3}{r^3} + \sigma_0 \left( 1 - \frac{r_0^3}{r^3} \right) - \frac{2\mu [b(p_1 - p_0) + 3\alpha K_0(T_1 - T_0)]}{\lambda_0 + 2\mu} \left( \frac{r_0}{r} - \frac{r_0^2}{r^2} \right) \]  \hspace{1cm} (309.24)
\[ \sigma_{\theta\theta} = \sigma_{\varphi\varphi} = p_1 \frac{r_0^3}{2r^3} - \sigma_0 \left(1 + \frac{r_0^3}{2r^3}\right) - \frac{\mu \left[b(p_1 - p_0) + 3\alpha K_0(T_1 - T_0)\right]}{\lambda_0 + 2\mu} \left[\frac{r_0}{r} \left(1 + \frac{r_0^2}{r^2}\right)\right] \]  

(309.25)

And the diffusion process can be achieved if that the time are large enough with respect to the characteristics diffusion time relative to point \( r \). When the boundary conditions for \( r = r_0 \) in fluid pressure and temperature which are \( p = p_1 \) and \( T = T_1 \) apply for the whole model, the following equations correspond to the undrained solution of the instantaneous drilling of a borehole in an infinite elastic medium.

\[ \xi^r = \frac{\sigma_0 + p_1 \frac{r_0^3}{r^3}}{4\mu} \quad \sigma_{rr} = -p_1 \frac{r_0^3}{r^3} + \sigma_0 \left(1 + \frac{r_0^3}{r^3}\right) \]

\[ \sigma_{\theta\theta} = \sigma_{\varphi\varphi} = p_1 \frac{r_0^3}{2r^3} + \sigma_0 \left(1 + \frac{r_0^3}{2r^3}\right) \]  

(309.26)

### 309.4.3 Discussion of the Results

The model is constructed as a quarter of a half ball. The cavity radius is 10 cm. As the outside boundary is fixed, to minimize the possibility of the sudden increase of the fluid bulk modulus, the outside radius of the sphere is set to be 2 m. The final mesh is generated as Fig.309.6. And the following boundary conditions apply: The nodes on \( XZ \) and \( YZ \) plane are fixed for solid and fluid displacement in \( Y \) and \( X \) direction respectively; the vertical solid and fluid displacement for the nodes on the \( XY \) plane are suppressed; for the nodes along the outside surface, to satisfy the complementary conditions, all the solid and fluid displacements are set to be zero as well. The pressure is translated in to nodal forces and applied in the radial direction. For simplicity, the hydrostatic stress \( \sigma_0 \) is equal to zero and with the assumption of thermodynamic equilibrium through the process, the temperature factor can be neglected. Also the initial fluid pressure \( p_0 \) is set to be 0 kPa. The analytical solution is studied below using the following set of parameters shown in Table 309.2.

As the same procedure in the previous drilling of borehole problem, we compared both the drained and undrained behavior. The drained and undrained behavior are tested by the permeability of \( k = 3.6 \times 10^{-4} m/s \) and \( k = 3.6 \times 10^{-8} m/s \) respectively. In drained behavior, we can see along the cavity surface, the experimental result of the radial displacement match the analytical solution very well. While with the increase of the radius, the decrease of the radial displacement for close solution is much smaller that of the experimental results. For the undrained behavior, we can see the radial displacement of the experimental results are always smaller than the close solution. Again it should be noted that the close solutions for the drained and undrained behavior along the cavity surface are exactly the same. This can be explained in the same way as the previous drilling of the borehole problem. When the experimental results from drained behavior are compared with the analytical undrained solution, it is observed they provide good agreement to each other as well.
### 309.5 Line Injection of a fluid in a Reservoir

#### 309.5.1 The Problem

Liquid water is usually injected into a reservoir from a primary well in order to recover the oil from a secondary well in petroleum engineering. This induces a problem of injecting a fluid into a cylindrical well of negligible dimensions.

Consider a reservoir of infinite extent composed of an isotropic, homogeneous and saturated poroelastic material. Through a cylindrical well of negligible dimensions, the injection of the same fluid is performed in all directions orthogonal to the well axis forming the $Oz$ axis of coordinates. As a result of the axisymmetry and cylindrically infinite, all quantities spatially depends on $r$ only. The injection starts at time $t = \Gamma$ and stops at time $t = \Gamma'$. The flow rate of fluid mass injection is constant and equal to

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson Ratio</td>
<td>$\nu$</td>
<td>0.2</td>
<td>-</td>
</tr>
<tr>
<td>Young's Modulus</td>
<td>$E$</td>
<td>1.2E+6</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Solid Bulk Modulus</td>
<td>$K_s$</td>
<td>3.6E+7</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Fluid Bulk Modulus</td>
<td>$K_f$</td>
<td>1.0E+17</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Solid Density</td>
<td>$\rho_s$</td>
<td>2.7</td>
<td>ton/m$^3$</td>
</tr>
<tr>
<td>Fluid Density</td>
<td>$\rho_f$</td>
<td>1.0</td>
<td>ton/m$^3$</td>
</tr>
<tr>
<td>Porosity</td>
<td>$n$</td>
<td>0.4</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 309.2: Material Properties used to study spherical cavity problem
309.5.2 Analytical Solution

This is a problem of cylindrically symmetry. Consequently the cylindrical coordinates \((r, \theta, z)\) is adopted. The vector of relative flow of fluid mass \(w\) reads

\[
w = w(r, t)e_r,\]

(309.27)

where \(e_r\) is the unit vector along the radius. Using the fluid mass balance relationship, it yields

\[
\int_0^r \frac{\partial m}{\partial t}(r, t)2\pi rdr = q - 2\pi rw(r, t) \quad \forall r, t
\]

(309.28)

In addition, we require the fluid flow to reduce to zero infinitely far from the well.
Based on above Eqs. 309.28-309.29, the radial displacement is derived in the form

\[ p = \frac{\Omega}{4\pi \rho_0^f kt} \exp\left(-\frac{r^2}{4c_m t}\right) \]

\[ \xi_r = \frac{bM\Omega}{2\pi \rho_0^f (\lambda + 2\mu)r} \left[ 1 - \exp\left(-\frac{r^2}{4c_m t}\right) \right] \]  \hspace{1cm} (309.30)

Using the constitutive equation, the stress field can be derived as follows:

\[ \sigma_{rr} = -2\mu \frac{\xi_r}{r} \hspace{0.5cm} \sigma_{\theta\theta} = 2\mu \frac{\xi_r}{r} - \frac{2\mu b}{\lambda_0 + 2\mu p} \]

\[ \sigma_{zz} = -\frac{2\mu b}{\lambda_0 + 2\mu a} p \]  \hspace{1cm} (309.31)
Figure 309.9: The comparison of radial solid displacement between analytical solution for undrained behavior and experimental result for drained behavior

309.5.3 Discussion of the Results

As a result of axisymmetry, the model can be constructed as a quarter of a pie. The radius of the pie is 1 m and the thickness of the pie is 5 cm. A cylindrical well is drilled at the center of the pie, and its radius is 1 cm, which can be neglected in dimension when compared with the whole pie. The final mesh is shown as Fig. 309.10. And the boundary conditions is as follows: As a consequence of plain strain problem, all the movements for solid and fluid in vertical direction $Oz$ are suppressed; the solid and fluid displacement for the nodes along the X axis and Y axis are fixed in Y and X direction respectively for the reason of axisymmetry; the nodes along the outside perimeter are fixed in the solid and fluid displacement with the assumption of infinite medium. To the difference with the previous problems, the traction boundary conditions are applied on the fluid displacement. It should be noted that the $\Omega$ mention in the above equations is the volume of the fluid injected per unit of vertical well length and has a unit of $m^3/m$. In order to generate the volume of 1 $cm^3/m$, the corresponding fluid displacement of the nodes along the well has been calculated and applied as a step function at the time of 0 sec. For simplicity, the initial fluid pressure $p_0$ is set to be 0 kPa. The analytical solution is studied below using the following set of parameters shown in Table 309.3.
Table 309.3: Material Properties used to study the line injection problem

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson Ratio</td>
<td>$\nu$</td>
<td>0.2</td>
<td>-</td>
</tr>
<tr>
<td>Young’s Modulus</td>
<td>$E$</td>
<td>1.2E+6</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Solid Bulk Modulus</td>
<td>$K_s$</td>
<td>3.6E+7</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Fluid Bulk Modulus</td>
<td>$K_f$</td>
<td>1.0E+17</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Undrained Bulk Modulus</td>
<td>$K_u$</td>
<td>6.0E+7</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Bulk Modulus</td>
<td>$K$</td>
<td>6.7E+5</td>
<td>kN/m$^2$</td>
</tr>
<tr>
<td>Solid Density</td>
<td>$\rho_s$</td>
<td>2.7</td>
<td>ton/m$^3$</td>
</tr>
<tr>
<td>Fluid Density</td>
<td>$\rho_f$</td>
<td>1.0</td>
<td>ton/m$^3$</td>
</tr>
<tr>
<td>Fluid Diffusivity coefficient</td>
<td>$c_f$</td>
<td>0.4973</td>
<td>m$^2$/s</td>
</tr>
<tr>
<td>Porosity</td>
<td>$n$</td>
<td>0.4</td>
<td>-</td>
</tr>
<tr>
<td>Permeability</td>
<td>$k$</td>
<td>3.6E-6</td>
<td>m/s</td>
</tr>
</tbody>
</table>

In the analysis, the pore pressure and the radial displacement are studied. The results are recorded from three points at the radius of 10 cm, 50 cm and 85 cm. The close solution and experimental results are shown in Fig.?? and Fig.?? As the time step is set to be 1 sec, the first data point starts at the time of 1 sec. From the pore pressure plot we can see that the build-up of the pore pressure reach the peak value of 34 kPa at the radius of 85 cm. With the decrease of the radius, the pore pressure decreases as well. This can be explained by the fact that the closer the point to the injection location, the earlier and the larger load is applied, so the pore pressure dissipates faster. And as time passes by, we can see...
the pore pressure progressively dissipates and finally almost reaches the same value within the model. The same phenomena can be been observed from the radial solid displacement. The maximum solid displacement occurs at the radius of 85 cm, which means more coupling between the solid and fluid phase, as consequence, the pore pressure should have the largest value. This corresponds to the previous result. With the increase of the time, the radial solid displacement get closer to zero, which means the fluid moves out the solid skeleton.

Figure 309.11: The comparison between analytical solution and experimental result for pore pressure
Figure 309.12: The comparison between analytical solution and experimental result for radial displacement
Figure 309.13: The comparison between analytical solution and experimental result for radial displacement
Table 309.4: Simulation parameters used for the shock wave propagation verification problem.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson ratio</td>
<td>$\nu$</td>
<td>0.3</td>
</tr>
<tr>
<td>Young’s modulus</td>
<td>$E$</td>
<td>$1.2 \times 10^6$ kN/m$^2$</td>
</tr>
<tr>
<td>Solid particle bulk modulus</td>
<td>$K_s$</td>
<td>$3.6 \times 10^7$ kN/m$^2$</td>
</tr>
<tr>
<td>Fluid bulk modulus</td>
<td>$K_f$</td>
<td>$2.17 \times 10^6$ kN/m$^2$</td>
</tr>
<tr>
<td>Solid density</td>
<td>$\rho_s$</td>
<td>2700 kg/m$^3$</td>
</tr>
<tr>
<td>Fluid density</td>
<td>$\rho_f$</td>
<td>1000 kg/m$^3$</td>
</tr>
<tr>
<td>Porosity</td>
<td>$n$</td>
<td>0.4</td>
</tr>
<tr>
<td>Newmark parameter</td>
<td>$\gamma$</td>
<td>0.6</td>
</tr>
</tbody>
</table>

309.6 Shock Wave Propagation in Saturated Porous Medium

In order to verify the dynamic behavior of the system, an analytic solution developed by Gajo (1995) and Gajo and Mongiovi (1995) for 1C shock wave propagation in elastic porous medium was used. A model was developed consisting of 1000 eight node brick elements, with boundary conditions that mimic 1D behavior. In particular, no displacement of solid ($u_x = 0, u_y = 0$) and fluid ($U_x = 0, U_y = 0$) in $x$ and $y$ directions is allowed along the height of the model. Bottom nodes have full fixity for solid ($u_i = 0$) and fluid ($U_i = 0$) displacements while all the nodes above base are free to move in $z$ direction for both solid and fluid. Pore fluid pressures are free to develop along the model. Loads to the model consist of a unit step function (Heaviside) applied as (compressive) displacements to both solid and fluid phases of the model, with an amplitude of 0.001 cm. The $u$–$p$–$U$ model dynamic system of equations was integrated using Newmark algorithm (see section 108.3). Table 309.4 gives relevant parameters for this verification.

Two set of permeability of material were used in our verification. The first model had permeability set $k = 10^{-6}$ cm/s which creates very high coupling between porous solid and pore fluid. The second model had permeability set to $k = 10^{-2}$ cm/s which, on the other hand creates a low coupling between porous solid and pore fluid. Comparison of simulations and the analytical solution are presented in Figure 309.14.

Before proceeding to the analysis, the following assumptions are made: For high-frequency components, the permeability remains constant; thus, the dependency of the permeability on the frequency is neglected. Unless specified, all the models in this report are elastic isotropic.
Figure 309.14: Compressional wave in both solid and fluid, comparison with closed form solution.

### 309.7 Vertical Consolidation of a soil layer by Coussy (2004)

#### 309.7.1 Brief review of Analytical Solution for Consolidation by Coussy (2004)

The consolidation process can be defined as follows: When an elastic soil layer is subjected to an external change in mean normal stress, immediately the water will alone sustain this increment of mean normal stress and cause the build-up the excessive pore water pressure. In the progress of the flow of the water to the surface, the load is gradually transferred to the soil skeleton and the excessive pore water pressure will dissipate. At the same time, the settlement of the soil layer occurs. As settlement is usually a major concern in geotechnical engineering, this is a key problem in soil mechanics.

Consider a soil layer composed of an isotropic, homogeneous and saturated thermoporoelastic material. The layer has a thickness of $h$ in the $Oy$ direction and of infinite extent in the two other directions $Ox$ and $Oy$. The layer is underlain by a rigid and impervious base at $y = 0$. And the top surface at $y = h$ is so perfectly drained that the pore pressure is held constant as zero.

At the initial state of the soil layer, the thermal effects are neglected so that the boundary conditions follow that:

\[ t \leq 0 \rightarrow \begin{align*}
    y &= h \\
    p &= 0 \\
    y &= 0 \\
    \frac{\partial p}{\partial z} &= 0
\end{align*} \] (309.32)

At time $t = 0$, tan instantaneous vertical load $-\varpi e_y$ is suddenly applied on the top surface $y = h$, 

\[ K = 10^{-6} \text{cm/s, FEM} \]

\[ K = 10^{-6} \text{cm/s, Closed Form} \]

\[ K = 10^{-2} \text{cm/s, FEM} \]

\[ K = 10^{-2} \text{cm/s, Closed Form} \]
the induced boundary conditions require that

\[ t > 0 \rightarrow y = h \quad \sigma_y = -\varpi y \] (309.34)

The undeformability of the substratum reads

\[ y = 0 \quad \xi = 0 \] (309.35)

The impermeability implies

\[ y = 0 \quad -w \cdot e_y = -w_y = 0 \] (309.36)

The problem is then to determine the new fields of fluid pressure, stress and displacement induced by the external loading.

Since this is a one-dimensional problem, the only non-zero displacement is the vertical displacement \( \xi_y \). But in particular the fluid pressure depends only on \( y \) and \( t \).

\[ \xi = \xi_y(y, t) e_y \quad p = p(y, t) \] (309.37)

Based on the constitutive equations from Coussy (2004), it follows that

\[ \sigma_{yy} = (\lambda_0 + 2\mu) \frac{\partial \xi_y}{\partial y} + bp \] (309.38)

\[ \sigma_{xx} = \sigma_{zz} = \frac{\lambda_0}{\lambda + 2\mu} \sigma_{yy} - \frac{2\mu b}{\lambda_0 + 2\mu} p \] (309.39)

And because the fluid pressure \( p \) must be an ordinary function of time \( t \), although the derivative of \( p \) is infinite at time \( t = 0 \) according to the consolidation equation (Coussy, 2004), the discontinuity of the fluid pressure \( p \) at time \( t = 0 \) must satisfy
\[ p(y, t = 0^+) = \eta \varpi \quad \varpi = \frac{\nu - \nu_0}{(1 - \nu)(1 - 2\nu_0)b} \]  

(309.40)

where \( \nu \) and \( \nu_0 \) are the drained and undrained Poisson ratio, respectively. For time \( t > 0 \), the vertical stress \( \sigma_{yy} = -\varpi \) is constant in time and space, therefore the diffusion equation reads

\[ \begin{align*}
 t > 0 & \quad c_m \frac{\partial^2}{\partial y^2} p = \frac{\partial}{\partial t} p \\
\end{align*} \]  

(309.41)

Collecting the above results, finally the fluid pressure reads

\[ p(y, t) = \eta \varpi \sum_{n=0}^{\infty} \frac{4(-1)^n}{\pi(2n + 1)} \cos\left[\frac{(2n + 1)\pi y}{2h}\right] \exp\left[-\frac{(2n + 1)^2\pi^2 t}{4\tau}\right] \]  

(309.42)

Each term of the series decreases exponentially with respect to the ratio \( \frac{t}{\tau} \), in which \( \tau \) is a characteristics consolidation time

\[ \tau = \frac{h^2}{c_m} \quad c_m = kM \frac{\lambda_0 + 2\mu}{\lambda + 2\mu} \]  

(309.43)

where \( \lambda \) and \( \lambda_0 \) are the drained and undrained Lame coefficient, respectively. Given by the Eqn.309.39, the only non-zero displacement \( \xi^y \) satisfies

\[ \begin{align*}
 \frac{\partial \xi^y}{\partial y} & = \frac{1}{\lambda_0 + 2\mu} (\sigma_{yy} + bp) \\
\end{align*} \]  

(309.44)

By substituting the value of \( -\varpi \) of the vertical stress and expression of (309.44), the series converges and it can integrated term by term yielding

\[ \xi^y(y, t) = \varpi \frac{y}{\lambda_0 + 2\mu} \left[ \frac{8nb}{\pi^2} \left\{ \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n + 1)^2} \sin\left[\frac{(2n + 1)\pi y}{2h}\right] \exp\left(-\frac{(2n + 1)^2\pi^2 t}{4\tau}\right) \right\} \right] \]  

(309.45)

(309.46)

Using Eqn.309.46 and substitute \( y = h \), the settlement can be expressed as

\[ \begin{align*}
 s(t) & = s_\infty + (s_{0^+} - s_\infty) \sum_{n=0}^{\infty} \frac{8}{\pi^2(2n + 1)^2} \exp\left(-\frac{(2n + 1)^2\pi^2 t}{4\tau}\right) \\
 s_{0^+} & = \frac{h\varpi}{\lambda + 2\mu} \quad s_\infty = \frac{h\varpi}{\lambda_0 + 2\mu} \\
\end{align*} \]  

(309.47)

(309.48)
309.7.2 Numerical Analysis

A soil column of ten brick \( u - p - U \) finite elements is used to model the horizontal layer. The height of the soil column is 10 m and the height of each element has dimensions \( 1m \times 1m \times 1m \), illustrated in Fig. 309.15. The material properties, shown in Table 309.5, are chosen as representative values for the natural soil deposit. Developed \( u - p - U \) finite element model, can simulate realistic compressibility of the pore fluid. However, it is important to note that the analytical solution for the vertical consolidation is based on the assumption that both the soil particles and the pore fluid (water) are completely incompressible. A uniform vertical pressure of 400 kPa is applied on the top surface of the soil column. The numerical analysis was performed in two stages:

(I) Self-Weight application (see Figures 309.15 to 309.19).

(II) Consolidation with drainage at top due to uniform vertical pressure of 400 kPa at the surface (see Figures 309.20 to 309.26).

Table 309.5: Material Properties used to study consolidation of a soil layer.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gravity acceleration</td>
<td>( g )</td>
<td>9.81 ( m/s^2 )</td>
</tr>
<tr>
<td>soil matrix Young’s Modulus</td>
<td>( E )</td>
<td>( 10 \times 10^3 ) kN/m(^2)</td>
</tr>
<tr>
<td>soil matrix Poisson’s ratio</td>
<td>( v )</td>
<td>0.25</td>
</tr>
<tr>
<td>solid particle density</td>
<td>( \rho_s )</td>
<td>( 2.65 \times 10^3 ) kg/m(^3)</td>
</tr>
<tr>
<td>water density</td>
<td>( \rho_f )</td>
<td>( 1.0 \times 10^3 ) kg/m(^3)</td>
</tr>
<tr>
<td>solid particle bulk modulus</td>
<td>( K_s )</td>
<td>( 37.0 \times 10^6 ) kN/m(^2)</td>
</tr>
<tr>
<td>fluid bulk modulus</td>
<td>( K_f )</td>
<td>( 2.2 \times 10^6 ) kN/m(^2)</td>
</tr>
<tr>
<td>porosity</td>
<td>( n )</td>
<td>0.46</td>
</tr>
<tr>
<td>Biot coefficient</td>
<td>( \alpha )</td>
<td>1.0</td>
</tr>
</tbody>
</table>

The following boundary conditions are applied to the model (Fig 309.27): As the bottom of the soil column is modeled as an undeformable and impermeable layer, both the solid and fluid displacements are fixed. The pore pressure is kept constant as zero at the top surface of the soil column because of the perfectly drained condition. In order to simulate the 1D consolidation problem, all the lateral movement of the solid and fluid phase are suppressed so that the vertical displacement is the only non-zero displacement for the intermediate nodes. To capture both the long term \((t > 0.1 \text{ sec})\) and short term \((t < 0.1 \text{ sec})\) response of the soil column, two different time steps are adopted: 0.1 sec and 0.005 sec, respectively. In order to observe the dissipation of the excessive pore water pressure in a reasonable and
Figure 309.15: Numerical Simulation of self weigh application and consolidation of a soil layer using u-p-U brick elements.

Convenient period, we select $k = 1.0 \times 10^{-4} m/s$ as the value for the permeability. To cure the artificial oscillation, some numerical damping is introduced into the analysis by using $\gamma = 0.6$ and $\beta = 0.3025$ for the Newmark integrator. Fig. 309.20 illustrates the physical geometry of the problem whereas Fig. 309.21 shows the numerical modeling.

Based on the above parameters, the other relative parameters can be calculated as follows:

The bulk modulus of the mixture:

$$K = \frac{E}{3(1-2\nu)} = 6.67 \times 10^3 kPa$$

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} = 6.67 \times 10^3 kPa \quad \mu = \frac{E}{2(1+\nu)} = 4 \times 10^3 kPa$$

The Biot coefficient:

$$b = 1 - \frac{K}{K_s} = 0.9998$$

The undrained bulk modulus of the mixture:

$$N = \frac{K_s}{b - n} = 6.85 \times 10^7 kPa \quad M = \frac{KfN}{Kf + Nn} = 4.47 \times 10^6 kPa$$
Figure 309.16: The distribution of the vertical effective stresses with depth due to self-weight. The stresses are calculated at the Gauss points within each brick element. The stresses obtained from the numerical analysis are equal for all the Gauss points within the element, due to the compatibility of deformations at the element interfaces.

\[ K_u = K + b^2 M = 4.4 \times 10^6 kPa \]  
(309.53)

The diffusion coefficient and characteristic time of consolidation:

\[ c_f = \frac{kM}{\gamma_w \left( K + \frac{4\mu}{3} \right)} = 1.2 m^2/s \quad t = \frac{h^2}{c_f} = 83.33 s \]  
(309.54)

### 309.7.3 Discussion of Numerical Results - Conclusions

The stage of self weight application shows in Fig. 309.19 that the expected estimated settlement is quite close to the one obtained from the analysis. The difference in the two values is due to the stress distribution coming out of the analysis which is slightly different than the one considered in theory (see Fig. 309.16) and possibly due to the compressibility of soil particles in the numerical analysis.

In Fig. 309.24, the normalized fluid pressure is plotted against the location for various normalized times. For normalized time \( T_v = 0.1 \), only the nodes close to the top free flow surface display the dissipation of the pore pressure. The experimental result provide good agreement with the analytical
solution. With the increase of the normalized time, we can clearly see the tendency of the dissipation of the water. At normalized time $T_v = 1.0$ (natural time $t = 83$ sec), the maximum normalized pore pressure is only about 0.11. It can be concluded that the numerical analysis can effectively demonstrate the process of the dissipation of the pore pressure.

In Fig. 309.26, the change of the porosity is predicted due to the consolidation of the soil layer. Both the change of volume of the soil ($0.1782 \text{m}^3$) and the fluid ($0.1794 \text{m}^3$) have been calculated. Theoretically, these two values should be the same, according to the fact that the settlement of the soil layer is due the fluid which is squeezed out, assuming that the soil grains are incompressible. However, this difference in the values is due to the compressibility of the soil grains, which was not considered infinite in the numerical analysis.

Figure 309.17: The distribution of the hydrostatic pore pressures, the effective and total stresses with depth, obtained from the numerical analysis after self-weight application.
Figure 309.18: Estimation of the total self load, as it is obtained from the numerical analysis, given the porosity and the densities of the fluid (ρ_f) and the grains (ρ_s or G_s).

### 309.8 One dimensional wave propagation in elastic porous media subjected to step displacement boundary condition

#### 309.8.1 Brief review of Analytical Solution by Gajo and Mongiovi (1995)

An one-dimensional exact analytical solution of the Biot’s equations is provided by Gajo and Mongiovi (1995) for the completely general solution of the transient problem in saturated, linear, elastic, porous media. The analytical solution was obtained was obtained by Fourier series. This solution is considered to be completely general because it is not based on any assumptions with respect to the inertial, viscous or mechanical coupling. Furthermore, it can be applied to any type of boundary-initial value problem.

The advantage of this analytical solution consists of showing the mechanics of dispersive wave propagation in saturated elastic solids. This is achieved by allowing the detailed analysis of wave fronts of the first and second kind of longitudinal waves and by analyzing accurately the effects of each term of coupling on the transient behavior of saturated porous media. In particular, since each term of the Fourier series represents a frequency component of the excitation signal, the analytical solution can
The total settlement due to self-weight can be calculated as follows:

\[
D = \frac{E (1-v)}{(1+v)(1-2v)}
\]

\[
D = 12 \text{ MPa}
\]

\[
S = \frac{\sigma' H}{D} = \frac{42 \times 10}{12000} = 0.035 \text{ m}
\]

The settlement from the analysis is 0.0363 m.

Figure 309.19: Prediction of the total settlement due to self-weight and comparison with the numerical result.

describe the behavior of each frequency component. Thus, it can illustrate the mechanics of dispersive wave propagation in which higher frequencies propagate with two waves and lower frequencies with only one wave, as a function of permeability and travel length.

Considering the above mentioned arguments, the analytical solution can provide a useful comparative term towards the verification and the validation of the existing numerical solutions based on the finite element method. Such a study was conducted by Gajo et al. (1994), by comparing analytical results with numerical ones obtained by a \( u - p - U \) numerical formulation.

In the paper by Gajo and Mongiovi (1995), the transient response of porous media is shown for typical material properties of a natural granular deposit and for different degrees of viscous coupling. Specifically, analytical results are given from the solution of the following one-dimensional boundary value problem: at the top and bottom surfaces of a soil layer of finite thickness \( L \), the excitation consisting of a step displacement boundary condition (Heaviside function) is applied to both solid and fluid phases. This problem can demonstrate better the mechanics of dispersive wave propagation, since the step excitation contains waves of all kind of frequencies. The analytical solution is relative only to the first arrival of the waves of the first and second kind.
CONSOLIDATION WITH SINGLE DRAINAGE AT TOP

\[ p_x = \sum_{m=1}^{\infty} \left[ \left( \frac{2P_0}{m\pi} \right) (1 - \cos m\pi) \sin \left( \frac{m\pi z}{2H_0} \right) \exp \left( -\frac{m^2 \pi^2 T}{4} \right) \right] \]

**Time Factor:** \[ T_v = C_v \frac{t}{H^2} \]

**Coefficient of Consolidation:**

\[ C_v = k D / Z_w \]

**In our case:**

\[ C_v = 10^{-3} \times \frac{12000}{10} = 1.2 \]

\[ t_{\text{total}} = \frac{H^2}{C_v} = \frac{100}{1.2} = 83.33 \text{ sec} \]

Figure 309.20: The physical geometry of the problem.
Boundary Conditions as before...

Nodal load at top ⇒ total = 400kN

Load

-100 kN

time

Newmark Integration Method
(β = 0.3025; γ = 0.6)

Dissipation of vibrations...

Figure 309.21: Numerical simulation of 1D consolidation of a soil layer.
Figure 309.22: Time history of the normalized excess pore pressure for three different normalized depths indicating faster dissipation close to the surface. The dissipation has practically been completed at $t = 83.33\sec$, as predicted.
Figure 309.23: Normalized excess pore pressures versus normalized time for three different normalized depths. Comparison of numerical results with the analytical ones.
Figure 309.24: Distribution of normalized excess pore pressures with normalized depth for four different time factors. Comparison of numerical results with the analytical ones.
Figure 309.25: Time history of the settlements for four different depths.
The vertical displacement of the fluid (upwards)\[\Rightarrow 0.39 \text{ m}\]

Volume of fluid that escaped = 46% \(0.39\) m
\[\Rightarrow 0.1794 \text{ m}^3\]

The settlement of the solid part from the analysis\[\Rightarrow 0.33 \text{ m}\]

Change of volume of soil = 54% \(0.33\) m\(^3\)
\[\Rightarrow 0.1782 \text{ m}^3\]

New Porosity, \(n = \frac{(4.6-0.1794)}{(10-0.1782)}\)

New Porosity, \(n = 0.45\)

Figure 309.26: Prediction of the change in porosity of the soil layer due to consolidation.
Boundary Conditions

- $u_y, u_x, U_y \text{ fixed at every node}$
- **Base**: all $u_i, U_i \text{ DOFs fixed, p free}$
- **Top**: $p \text{ fixed (p = 0)}$

**Figure 309.27**: Boundary Conditions applied to the numerical model throughout the self-weight application and the process of consolidation due to extra load applied to the surface of the soil column.
309.8.2 Numerical Analysis

Numerical examples for three different values of viscous coupling \( k = 10^{-8} \text{cm}^3/\text{s}/\text{g}, k = 10^{-6} \text{cm}^3/\text{s}/\text{g}, k = 10^{-5} \text{cm}^3/\text{s}/\text{g} \) were solved in order to verify the previously mentioned \( u-p-U \) formulation in a wide range of drag. The numerical model used for the simulation of the 1C shock wave propagation consists of 400 u-p-U brick finite elements of dimensions \( 0.01 \text{cm} \times 0.01 \text{cm} \times 0.01 \text{cm} \) creating a soil column 4 cm thick. Figure 309.28 illustrates the transition from the physical configuration of the problem to its numerical simulation. Table 309.6 shows the soil properties of the numerical model.

Figure 309.28: The numerical model used for the verification of the finite element implementation through comparison with the analytical results provided by Gajo and Mongiovi (1995).

At the top surface of the soil column, a step displacement of \( 1.0 \times 10^{-3} \text{cm} \) is applied both to the solid and the fluid phase. Only the vertical displacement is free. There is no lateral flow or displacement. The degree of freedom related to the pore pressures is constrained at the top surface to be equal to the atmospheric pressure and is free at the rest of the nodes. The base of the model is rigid and impervious.

This kind of excitation (Heaviside function) applied at the top of the model, results clearly in waves of all kinds of frequency, first due to its nature and secondarily due to the way of its application. This fact together with the great stiffness of the solid skeleton (see Table 309.6 require a very dense spatial
Table 309.6: Soil Properties for 1C shock wave propagation for the problem by Gajo and Mongiovi (1995).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gravity acceleration</td>
<td>$g$</td>
<td>$9.81 , m/s^2$</td>
</tr>
<tr>
<td>soil matrix Young’s Modulus</td>
<td>$E$</td>
<td>$1.2 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>soil matrix Poisson’s ratio</td>
<td>$v$</td>
<td>0.3</td>
</tr>
<tr>
<td>solid particle density</td>
<td>$\rho_s$</td>
<td>$2.7 \times 10^3 , kg/m^3$</td>
</tr>
<tr>
<td>water density</td>
<td>$\rho_f$</td>
<td>$1.0 \times 10^3 , kg/m^3$</td>
</tr>
<tr>
<td>solid particle bulk modulus</td>
<td>$K_s$</td>
<td>$36.0 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>fluid bulk modulus</td>
<td>$K_f$</td>
<td>$2.177 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>porosity</td>
<td>$n$</td>
<td>0.4</td>
</tr>
<tr>
<td>Biot coefficient</td>
<td>$\alpha$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Discretization. Here, 400 u-p-U brick finite elements of dimensions $0.01 cm \times 0.01 cm \times 0.01 cm$, following similar discretization with Gajo et al. (1994). The time step, $\delta t$ required needs to be limited to

$$\delta t < \frac{\delta h}{v}$$  \hspace{1cm} (309.55)

$$\delta t < \frac{\delta h}{v}$$  \hspace{1cm} (309.56)

where $v$ is the highest wave velocity. In our case, the temporal integration involves 800 steps of $2.0 \times 10^{-8} \, sec$, which allows a maximum wave velocity of $5.0 \times 10^5 \, m/s$.

Two different time integration methods were used: i) The Newmark integrator and ii) The Hilber-Hughes-Taylor (HHT) Integrator. Sets of parameters, assuring unconditionally numerical stability, were chosen for both integrators. For the case of Newmark integrator (see Figures 309.29 - 309.44), the following sets of parameters were used:

a) $\gamma = 0.5$ and $\beta = 0.25$,

b) $\gamma = 0.6$ and $\beta = 0.3025$,

c) $\gamma = 0.7$ and $\beta = 0.4$. 
Figure 309.29: Time history of solid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-8} \text{cm}^3/\text{s}/\text{g}$). Two different sets of Newmark parameters were used for the numerical analysis.
Figure 309.30: A magnified view of Figure 309.29 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.31: Time history of fluid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-8} \text{cm}^3\text{s}/\text{g}$). Two different sets of Newmark parameters were used for the numerical analysis.
Figure 309.32: A magnified view of Figure 309.31 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.33: Time history of solid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-6} \text{cm}^3\text{s/g}$). Two different sets of Newmark parameters were used for the numerical analysis.
Figure 309.34: A magnified view of Figure 309.33 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.35: Time history of fluid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-6} \text{cm}^3 \text{s/g}$). Two different sets of Newmark parameters were used for the numerical analysis.
For the HHT integrator (see Figures 309.45 - 309.56), the following sets of parameters were used:

a) $\alpha = -0.1$, $\gamma = 0.6$ and $\beta = 0.3025$,

b) $\alpha = -0.3$, $\gamma = 0.8$ and $\beta = 0.4225$. 

Figure 309.36: A magnified view of Figure 309.35 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.37: Time history of solid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling \(k = 10^{-5} \text{cm}^3/\text{s}/\text{g}\). Two different sets of Newmark parameters were used for the numerical analysis.
Figure 309.38: A magnified view of Figure 309.37 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.39: Time history of fluid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-5}\text{ cm}^3\text{s/g}$). Two different sets of Newmark parameters were used for the numerical analysis.
Figure 309.40: A magnified view of Figure 309.39 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.41: Time history of solid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for three different values of viscous coupling. The Newmark set of parameters used for the numerical solution was: $\gamma = 0.6$ and $\beta = 0.3025$. 
Figure 309.42: A magnified view of Figure 309.41 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.43: Time history of fluid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for three different values of viscous coupling. The Newmark set of parameters used for the numerical solution was: $\gamma = 0.6$ and $\beta = 0.3025$. 
Figure 309.44: A magnified view of Figure 309.43 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.45: Time history of solid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-8}$ cm$^3$/s/g). Two different sets of unconditional stable HHT parameters were used for the numerical analysis.
Figure 309.46: A magnified view of Figure 309.45 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.47: Time history of fluid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-8} \text{cm}^3\text{s/g}$). Two different sets of unconditional stable HHT parameters were used for the numerical analysis.
Figure 309.48: A magnified view of Figure 309.47 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.49: Time history of solid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling\((k = 10^{-6}\text{cm}^3\text{s/g})\). Two different sets of unconditional stable HHT parameters were used for the numerical analysis.
Figure 309.50: A magnified view of Figure 309.49 illustrating the details of wave front of the longitudinal wave of first kind.

\[ k = 10^{-8} \text{ cm}^3 \text{s/g} \]

\[ t \left[ 10^{-6} \text{ sec} \right] \]

solid displacement (cm)

FEM (HHT, alpha=0.1)
FEM (HHT, alpha=0.3)
Analytical Solution
Figure 309.51: Time history of fluid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling ($k = 10^{-6}$ cm$^3$s/g). Two different sets of unconditional stable HHT parameters were used for the numerical analysis.
Figure 309.52: A magnified view of Figure 309.51 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.53: Time history of solid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling \((k = 10^{-5}\text{cm}^3\text{s/g})\). Two different sets of unconditional stable HHT parameters were used for the numerical analysis.
Figure 309.54: A magnified view of Figure 309.53 illustrating the details of wave front of the longitudinal wave of first kind.
Figure 309.55: Time history of fluid displacements of longitudinal waves at 1 cm below the surface. Comparison of numerical results (FEM) with the analytical solution by Gajo and Mongiovi (1995) for the case of viscous coupling($k = 10^{-5}\text{ cm}^3\text{s/g}$). Two different sets of unconditional stable HHT parameters were used for the numerical analysis.
Figure 309.56: A magnified view of Figure 309.55 illustrating the details of wave front of the longitudinal wave of first kind.
309.8.3 Discussion of Numerical Results - Conclusions

Biot has shown that when dissipation is present, each frequency component propagated with its own velocity. Thus, especially in the case of numerical solutions using a finite element procedure, the response is very sensitive to the numerical damping introduced to the system. Generally, a drawback of all types of numerical solutions is the distortion and the smearing of the wave fronts, which are linked to the highest frequency that is allowed by the computational grid and the numerical damping due to the time integration method. The numerical results presented here, show a larger rise time than the analytical solution, that it could potentially be improved by using a finer spatial and temporal discretization.

In particular, the dissipation of high frequency oscillations is achieved more efficiently by the Newmark integrator than the HHT one. Due to the fact that the filtering of high frequencies is less in case of HHT integrator, the smearing of the wave front of the first kind of longitudinal waves is not so extensive as in the Newmark case. Obviously, for both cases, as the numerical damping increases by changing the sets of parameters, the rise time of the water fronts increases too. It is also worth mentioning that the rise time of the wave front of the second kind low-frequency longitudinal wave is even longer than that of the first kind.

Figures 309.41 and 309.44 illustrate the comparative results for all the three different values of viscous coupling using the Newmark integration method. In general, it is worth noting that the numerical results are in good agreement with the main characteristics of the mechanics of dispersive wave propagation in fully saturated, porous media, as indicated by the analytical results. For example, numerical results well demonstrate that during the propagation of the first wave, the solid and fluid displacement are in phase with each other, whereas during the propagation of second wave, the displacements of the two phases are in opposition. Overall, the finite element solutions reproduce correctly the forms of wave propagation for a wide range of permeability.

309.9 One dimensional wave propagation in elastic porous media subjected to step loading at the surface

309.9.1 Brief review of Analytical Solution by de Boer et al. (1993)

An analytical solution for the one-dimensional transient wave propagation fluid-saturated elastic porous media is provided by de Boer et al. 1993. The fluid-saturated porous material is modeled as a two phase system composed of an incompressible solid phase and an incompressible fluid phase. An exact analytical solution is obtained via Laplace transform technique considering initial and boundary conditions, which exhibits only one independent compressive wave in both the solid and fluid phases, as a result of the
incompressibility constraint.

The problem configuration, which the analytical solution is addressed to, consists of an one-dimensional infinitely long column, separated from the half-space of a fluid-saturated porous elastic skeleton material. The motion of both the solid and the fluid materials is constrained to occur in the vertical direction. Loading as a function of time, \( \sigma(z=0,t) = f(t) \), is applied to the half space surface boundary by a permeable punch with ideal permeability. Homogeneous pore distribution and free pore fluid surface are assumed. The wave motion in the porous medium is expressed by the solid and fluid displacements or the solid extra stresses, respectively, but it cannot be expressed by the pore pressure which is just the Lagrangian multiplier corresponding to the incompressibility constraint of the medium.

In particular, in the paper by de Boer et al. (1993), the solid and the fluid displacements, the solid skeleton extra stresses and the pore pressure are given with respect to time and with respect to different depths in the soil column within the framework of three loading forms: i) sinusoidal, ii) step loading and iii) impulsive loading. These results can be taken for a quantitative comparison to various numerical solutions.

**309.9.2 Numerical Analysis**

Numerical example for the step loading case was solved in order to verify the previously mentioned \( u - p - U \) formulation. The numerical model used for the simulation of the 1C shock wave propagation consists of 1000 u-p-U brick finite elements of dimensions 1cm \( \times \) 1cm \( \times \) 1cm creating a soil column 10m thick. Obviously, the numerical simulation of a semi infinite soil column is not possible; thus, a soil column of thickness of 10 cm was considered adequate for the current problem configuration. Figure 309.57 illustrates the transition from the physical configuration of the problem to its numerical simulation. Table 309.7 shows the soil properties of the numerical model, which are the same with those used for the analytical results presented in the paper by de Boer et al. (1993).

The only difference is noted on the elastic modulus, which was selected to be 20\( MN/m^2 \) for the numerical solution (FEM) instead of 30\( MN/m^2 \), as it is mentioned in the previously mentioned paper. This is due to the fact that the numerical results indicated that the results given in the paper correspond to a soil column with elastic modulus equal to 20\( MN/m^2 \) instead of 30\( MN/m^2 \). Moreover, it should be mentioned that the solid and fluid compressibility were given realistic values (see Table 309.7), which practically means that the two constituents are incompressible.

At the top surface of the soil column, a step loading of \( \sigma(z=0,t) = 3kN/m^2 \) is applied to the solid part, as a nodal load equally distributed to the four top nodes. The nodal load is expressed as:
The vertical displacement is free. There is no lateral flow or displacement. The degree of freedom related to the pore pressure is constrained at the top surface to be equal to the atmospheric pressure, while it is free at the rest of the nodes. The base of the model is rigid and impervious.

This kind of excitation (Heaviside function) applied at the top of the model, results clearly in waves of all kinds of frequency, first due to its nature and secondarily due to the way of its application. Due to this fact, a fairly dense mesh of 1000 u-p-U brick finite elements of dimensions $1\,\text{cm} \times 1\,\text{cm} \times 1\,\text{cm}$ was chosen. The time step, $\delta t$ required needs to be limited to

\begin{equation}
\delta t < \frac{\delta h}{v} \tag{309.58}
\end{equation}

\begin{equation}
\delta t < \frac{\delta h}{v} \tag{309.59}
\end{equation}

\begin{equation}
F_N(z = 0, t) = \frac{\sigma(z = 0, t) \times A}{4} = \frac{3\,\text{kN/m}^2 \times 0.01\,\text{m} \times 0.01\,\text{m}}{4} = 7.5 \times 10^{-5}\,\text{kN} \tag{309.57}
\end{equation}
Table 309.7: Soil Properties for 1C shock wave propagation for the problem by de Boer et al. (1993)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gravity acceleration</td>
<td>$g$</td>
<td>$9.81 , m/s^2$</td>
</tr>
<tr>
<td>soil matrix Young’s Modulus</td>
<td>$E$</td>
<td>$20 \times 10^3 , kN/m^2$</td>
</tr>
<tr>
<td>soil matrix Poisson’s ratio</td>
<td>$v$</td>
<td>0.2</td>
</tr>
<tr>
<td>soil matrix Lame’s constant</td>
<td>$\lambda$</td>
<td>$5.55 \times 10^3 , kN/m^2$</td>
</tr>
<tr>
<td>soil matrix shear modulus</td>
<td>$\mu$</td>
<td>$8.33 \times 10^3 , kN/m^2$</td>
</tr>
<tr>
<td>solid particle density</td>
<td>$\rho_s$</td>
<td>$2.0 \times 10^3 , kg/m^3$</td>
</tr>
<tr>
<td>water density</td>
<td>$\rho_f$</td>
<td>$1.0 \times 10^3 , kg/m^3$</td>
</tr>
<tr>
<td>solid particle bulk modulus</td>
<td>$K_s$</td>
<td>$36.0 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>fluid bulk modulus</td>
<td>$K_f$</td>
<td>$2.177 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>porosity</td>
<td>$n$</td>
<td>0.33</td>
</tr>
<tr>
<td>Darcy’s permeability</td>
<td>$k_D$</td>
<td>0.01 m/s</td>
</tr>
</tbody>
</table>

where $v$ is the highest wave velocity. In our case, the temporal integration involves 4000 steps of $1.0 \times 10^{-4} \, sec$, which allows a maximum wave velocity of $100 \, m/s$. The propagation velocity can be calculated by the following equation given by de Boer et al. (1993) and is equal to $90.7 \, m/s$.

$$v = \sqrt{\frac{n^2(\lambda + 2\mu)}{n^2(1-n)\rho_s + (1-n)^2(n\rho_f)}} = 90.7 \, m/s \tag{309.60}$$

The Newmark time integration method was used, which dissipates more efficiently the high frequencies introduced in the system due to numerics than HHT integrator, as shown in section 4.3. The following set of parameters was chosen, assuring unconditionally numerical stability: $\gamma = 0.7$ and $\beta = 0.4$.

309.9.3 Discussion of Numerical Results - Conclusions

Figures 309.58 to 309.67 illustrate the comparative results between analytical and numerical solution. In general, it is worth noting that the numerical results are in good agreement, with respect to time and with respect to depth, with those obtained by the analytical solution. The responses of the medium due to step loading are indicative of the consolidation process in case of a free pore water surface. The solid moves downwards, indicating that settlement occurs and the fluid is squeezed out from the pore volume creating an upward flow. During the consolidation process, the extra solid skeleton stresses increase with
time at a certain depth. However, they decrease with the distance from the loading surface at a certain time. In opposition to the extra solid skeleton stresses, the pore pressure decreases with time tending to zero, while it increases with depth.

![Graph](image_url)

Figure 309.58: Time history of solid displacements at different depths due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).

Overall, it is worth mentioning that the results obtained from the finite element procedure practically coincide with the ones given by the analytical solution. The only difference is located to the pore pressure (see Figures 309.64 to 309.67), where numerical response is oscillatory in contrast to the analytical solution. This may be due to the high frequencies introduced to the system by the temporal and spatial discretization and/or the compressibility of the solid and fluid phases. It should be mentioned again, that in the analytical solution, the two constituents are assumed incompressible whereas in the numerical model, the solid and fluid bulk moduli have realistic values (see Table 309.7. That is why in Figure 309.68, numerical examples with different values of fluid compressibility were solved. It is obvious that the oscillations decrease as the fluid becomes more and more compressible because the stiffness of the system decreases and the high frequencies are dissipated faster. Moreover, Figure 309.68 indicated that better quantitative agreement between the numerical and analytical solution is achieved when the bulk modulus of the fluid is $2.2 \times 10^6$ kPa - realistic value - instead of $2.2 \times 10^9$ kPa, which would be expected since the pore fluid is assumed to be incompressible in the framework of the analytical solution.
Comparing the pore pressures obtained from these two cases, it can be observed that the pore pressure generation in the more compressible fluid is slightly higher than that of the almost incompressible fluid because of the existence of the oscillatory waves, as mentioned by (Zienkiewicz and Shiomi, 1984).

309.10 One dimensional wave propagation in elastic porous media subjected to step velocity boundary condition

309.10.1 Brief review of Analytical Solution by Hiremath et al. (1988)

Hiremath et al. (1988) present a solution of Biot's dynamic equation of motion for one-dimensional wave propagation in a fluid-saturated linear elastic isotropic soil using Laplace transformation followed by numerical inversion. This study is considered to be an extension of the exact transient solution presented by Garg et al. (1974) for two limiting cases of infinitely small and infinitely large viscous coupling. In both cases, a soil column of finite dimension subjected to velocity boundary conditions was analyzed, allowing for reflection of waves at the boundaries.

In particular, Hiremath et al. (1988) examines two cases allowing for weak and strong viscous coupling,
or else, as it is referred in the related paper, low and high drag, respectively. Moreover, two different types of excitations were applied at the boundary surface in terms of solid and fluid velocity. In the first case, a unit step boundary condition boundary condition was applied at the top surface for both solid and fluid phases. In the second case, the fluid velocity specified at the boundary is different from the specified solid velocity increasing gradually to unity over the time scale. The results obtained from the numerical inversion allowed for six reflections of the fast compressional wave of first kind and two reflections of the secondary slow longitudinal wave.

One of the most important observations which both Garg et al. (1974) and Hiremath et al. (1988) concluded to, is that in case of strong viscous coupling (high drag), the material behaves as a single continuum with internal dissipation and the two wave fronts tend to become a single one.

Hiremath et al. (1988) presented a comparison of finite element solution of Biot’s equations of motion with one based on numerical inversion of the Laplace transform solution. It is explained in the paper that a proper choice of element type and time domain integration is essential for capturing the results coming from the semi-analytical solution. Moreover, the spatial discretization needs to be combined with an appropriate temporal one, so that the wave does not traverse more than one element length during a single time step. In detail, Hiremath et al. (1988) suggests 100 linear elements of 0.005 m length and

![Graph](image-url)

Figure 309.60: Time history of fluid displacements at different depths due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).
Figure 309.61: Response of fluid displacements versus depth at different time moments due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).

986 time steps of size $10^{-6}$ sec.

### 309.10.2 Numerical Analysis

Numerical examples for two extreme values of viscous coupling: a) high drag ($k = 0.148 \times 10^{-8} cm^3/s/g$) and b) low drag ($k = 0.148 \times 10^{-2} cm^3/s/g$), were solved in order to verify the previously mentioned $u - p - U$ formulation by comparing the results with the semi-analytical solution provided by Hiremath et al. (1988). The numerical model used for the simulation of the 1C shock wave propagation consists of 100 u-p-U brick finite elements of dimensions $0.005m \times 0.005m \times 0.005m$ creating a soil column 50cm thick. Figure 309.69 illustrates the transition from the physical configuration of the problem to its numerical simulation. Table 309.8 shows the soil properties of the numerical model.

At the top surface of the soil column, a step velocity of $1.0 \times 10^{-2} m/sec$ is applied both to the solid and the fluid phase. Only the vertical translational degrees of freedom are free. The horizontal translational degrees of freedom are constrained so that there is no lateral flow or displacement. The base of the model is rigid and impervious.

This kind of excitation (Heaviside function) applied at the top of the model, results clearly in waves of all kinds of frequency, first due to its nature and secondarily due to the way of its application. This
Figure 309.62: Time history of solid skeleton stresses at different depths due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).

Table 309.8: Soil Properties for 1C shock wave propagation for example by Hiremath et al. (1988).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gravity acceleration</td>
<td>$g$</td>
<td>$9.81 , m/s^2$</td>
</tr>
<tr>
<td>soil matrix Young’s Modulus</td>
<td>$E$</td>
<td>$23.21 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>soil matrix Poisson’s ratio</td>
<td>$v$</td>
<td>0.171</td>
</tr>
<tr>
<td>solid particle density</td>
<td>$\rho_s$</td>
<td>$2.66 \times 10^2 , kg/m^3$</td>
</tr>
<tr>
<td>water density</td>
<td>$\rho_f$</td>
<td>$1.0 \times 10^3 , kg/m^3$</td>
</tr>
<tr>
<td>solid particle bulk modulus</td>
<td>$K_s$</td>
<td>$36.0 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>fluid bulk modulus</td>
<td>$K_f$</td>
<td>$2.2 \times 10^6 , kN/m^2$</td>
</tr>
<tr>
<td>porosity</td>
<td>$n$</td>
<td>0.18</td>
</tr>
<tr>
<td>Biot coefficient</td>
<td>$\alpha$</td>
<td>0.6772</td>
</tr>
</tbody>
</table>
Figure 309.63: Response of solid skeleton stresses versus depth at different time moments due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).

Fact together with the great stiffness of the solid skeleton (see Table 309.8) requires a very dense spatial discretization. Here, 100 u-p-U brick finite elements of dimensions $0.005m \times 0.005m \times 0.005m$ were chosen, following similar discretization with Hiremath et al. (1988). The time step, $\delta t$, required needs to be limited to

$$\delta t < \frac{\delta h}{v}$$

(309.61)

$$\delta t < \frac{\delta h}{v}$$

(309.62)

where $v$ is the highest wave velocity. In our case, the temporal integration involves 1972 steps of $5.0 \times 10^{-7} sec$, in comparison with 986 time steps of size $10^{-6} sec$, used by Hiremath et al. (1988). The time integration method used was the Newmark integrator with parameters: $\gamma = 0.6$ and $\beta = 0.3025$.

**309.10.3 Discussion of Numerical Results - Conclusions**

Biot has shown that when dissipation is present, each frequency component propagates with its own velocity. Thus, especially in the case of numerical solutions using a finite element procedure, the response
is very sensitive to the numerical damping introduced to the system. Generally, a drawback of all types of numerical solutions is the distortion and the smearing of the wave fronts, which are linked to the highest frequency that is allowed by the computational grid and the numerical damping due to the time integration method. The numerical results presented here, show some oscillations at the rough changes in velocity due to reflection of wave fronts that could be possibly diminished by using a finer spatial and temporal discretization.

Figures 309.70 to 309.77 illustrate the comparative results for both extreme cases of viscous coupling. In general, it is worth noting that the numerical results are in good agreement with the main characteristics of the mechanics of dispersive wave propagation in fully saturated, porous media, as indicated by the semi-analytical results. For example, numerical results well demonstrate that for the case of strong viscous coupling (high drag), the solid and fluid are in phase with each other, implying that the two-phase material behaves as a single continuum. Overall, the finite element solutions reproduce correctly the trends of wave propagation in both limiting cases of viscous coupling.
Figure 309.65: Response of pore pressure versus depth at \( t = 0.01 \text{ sec} \) due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).
Figure 309.66: Response of pore pressure versus depth at $t = 0.05\, \text{sec}$ due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).
Figure 309.67: Response of pore pressure versus depth at $t = 0.1\, sec$ due to step loading. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).
Figure 309.68: Time history of pore pressure at 1 m below the ground surface due to step loading for different fluid compressibility. Comparison of numerical results (FEM) with the analytical solution by de Boer et al. (1993).
$v_s(0,t) = v_f(0,t) = 1 \text{ cm/s}$

Excitation at top: Heaviside function

Fully saturated soil column.
No lateral flow.
No lateral displacements.

100 brick elements

Figure 309.69: The numerical model used for the verification of the finite element implementation through comparison with the semi-analytical results provided by Hiremath et al. (1988).
Figure 309.70: Time history of solid velocity at 10 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-8} \text{cm}^3 \text{s/g}$).
Figure 309.71: Time history of solid velocity at 30 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-8} \text{cm}^3 \text{s/g}$).
Figure 309.72: Time history of fluid velocity at 10 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-8} cm^3 s/g$).
Figure 309.73: Time history of fluid velocity at 30 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-8} \text{cm}^3/\text{s/g}$).
Figure 309.74: Time history of solid velocity at 10 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-2} cm^3 s/g$).
Figure 309.75: Time history of solid velocity at 30 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-2} \text{cm}^3 \text{s/g}$).
Figure 309.76: Time history of fluid velocity at 10 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-2} cm^3 s/g$).
Figure 309.77: Time history of fluid velocity at 30 cm below the surface. Comparison of numerical results (FEM) with the semi-analytical solution by Hiremath et al. (1988) for the case of high drag ($k = 0.148 \times 10^{-2} \text{cm}^3 \text{s/g}$).
Chapter 310

Verification and Validation for Seismic Wave Propagation Problems

(In collaboration with Dr. Nima Tafazzoli, Dr. Matthias Preisig, Dr. Federico Pisanò, Mr. Kohei Watanabe, Mr. Chao Luo, and Dr. Hexiang Wang)
310.1 Chapter Summary and Highlights

310.2 Wavelet Seismic Signals

A wavelet is a wave with specific definitions and parameters. The amplitude of a wavelet usually starts at zero, increases by time and ended up at zero again. Typically a wavelet can be plotted as a brief oscillation such as a the small oscillation recorded by seismogram. There are different types of wavelets each with their own properties used for specific purpose in signal processing. For specific purposes different wavelets might be summed up to come up with new type of wave. A recently developed wavelet analysis has become a powerful tool to analyze the soil-structure systems for transient loads providing information both in time and frequency domains. In wavelet representation the basis functions are localized and contained in finite time domains (Sarica and Rahman (2003)).

310.2.1 Ricker Wavelet

One type of wavelet motions is the Ricker wave (Ryan (1994), Mavroeidis and Papageorgiou (2003)). The formulation of Ricker wavelet is shown in Equations (310.1):

\[ R(t) = A \left( 1 - 2\pi^2 f^2 t^2 \right) \exp(-\pi^2 f^2 t^2) \]  \hspace{1cm} (310.1)

where \( R(t) \) is the amplitude of the function in time, \( A \) if the maximum amplitude, and \( f \) is the peak frequency on the wavelet’s frequency spectrum. Figure (310.1) shows the actual time history and fast Fourier transform of Ricker wavelet, where \( A \) is taken as 1 and \( f \) is taken as 5 Hz. As it is shown, the frequency range of the motion is narrower compared to the real earthquake motion.

![Figure 310.1: Frequency content and a time domain representation of Ricker wavelet](image-url)
310.2.2 Ormsby Wavelet

Another example of interesting wavelet is called Ormsby wavelet (Ryan (1994)) which features a controllable flat frequency content with formulation shown in Equation (310.2).

\[
\begin{align*}
f(t) &= A \left( \left( \frac{\pi f_1^2}{f_4 - f_3} \text{sinc}(\pi f_4(t - t_s)) \right)^2 - \left( \frac{\pi f_3^2}{f_4 - f_3} \text{sinc}(\pi f_3(t - t_s)) \right)^2 \right) \\
&\quad - \left( \frac{\pi f_2^2}{f_2 - f_1} \text{sinc}(\pi f_2(t - t_s)) \right)^2 - \left( \frac{\pi f_1^2}{f_2 - f_1} \text{sinc}(\pi f_1(t - t_s)) \right)^2 \right) 
\end{align*}
\] (310.2)

where \( f_1 \) and \( f_2 \) define the lower range frequency band, \( f_3 \) and \( f_4 \) define the higher range frequency band, \( A \) is the amplitude of the function, and \( t_s \) is the time that maximum amplitude is happening, and \( \text{sinc}(x) = \frac{\sin(x)}{x} \).

Figure (310.2) shows an example of Ormsby wavelet in time domain and frequency domain. In this case, wave has a flat frequency range of 5Hz to 20Hz. Shown in Figure (310.3) is half of the Ormsby wavelet in frequency domain which the frequency range starts from 0 and remains constant up to 20Hz. This type of motion could be useful when low frequency range of motions are required for dynamic analysis of the systems.

Such broad band signals could be used to assess different aspects of soil-structure systems and with different incoming wave inclinations. While wavelet time domain motions are not the same as actual earthquakes, the idea is to use them for dynamic analysis of soil-structure systems for possible problems coming out of dynamic behavior, at different frequencies and for different energy input levels.

When used with the DRM, motions developed from different directions, different incident angles and different energies, will create a full envelope of these motions, which then can be used to evaluate performance based response of the soil-structure systems.
Figure 310.2: Frequency content and a time domain representation of an Ormsby wavelet, with constant frequency content between 5Hz and 20Hz.

Figure 310.3: Frequency content and a time domain representation of half of Ormsby wavelet formulation, with minimum frequency of zero and maximum of 20Hz.
310.3 Finite Element Mesh Size Effects on Seismic Wave Propagation Modeling and Simulation

310.3.1 Analysis Cases

Summary of the cases is shown in Table below. the input motion used is Ormsby wavelet which the corner cutoff frequency is shown in the table.

<table>
<thead>
<tr>
<th>Case Number</th>
<th>Model Height (m)</th>
<th>Shear Wave Velocity (m/s)</th>
<th>Element Size (m)</th>
<th>Frequency Cutoff (Hz)</th>
<th>Maximum Propagation Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>1000</td>
<td>10</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>1000</td>
<td>20</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>1000</td>
<td>10</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>1000</td>
<td>20</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>1000</td>
<td>50</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>1000</td>
<td>50</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>100</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>100</td>
<td>2</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>100</td>
<td>10</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1000</td>
<td>100</td>
<td>10</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1000</td>
<td>100</td>
<td>20</td>
<td>8</td>
<td>0.5</td>
</tr>
<tr>
<td>14</td>
<td>1000</td>
<td>100</td>
<td>50</td>
<td>8</td>
<td>0.2</td>
</tr>
</tbody>
</table>

310.3.2 Comparison of Case 1 and 2
Figure 310.4: Displacement time history of input motion (Ormsby Wavelet)

Figure 310.5: Comparison of displacement time histories of case 1 and 2 at top of the model
Figure 310.6: Comparison of FFT of case 1 and 2 at top of the model and input motion at the bottom of model

310.3.3 Comparison of Case 3 and 4

310.3.4 Comparison of Cases 3, 4, and 6

310.3.5 Comparison of Case 7 and 8
Figure 310.7: Displacement time history of input motion (Ormsby Wavelet)

Figure 310.8: Comparison of displacement time histories of case 3 and 4 at top of the model
Figure 310.9: Comparison of FFT of case 3 and 4 at top of the model and input motion at the bottom of model

Figure 310.10: Displacement time history of input motion (Ormsby Wavelet)
Figure 310.11: Comparison of displacement time histories of case 3, 4, and 6 at top of the model

Figure 310.12: Comparison of FFT of case 3, 4, and 6 at top of the model and input motion at the bottom of model
Figure 310.13: Displacement time history of input motion (Ormsby Wavelet)

Figure 310.14: Comparison of displacement time histories of case 7 and 8 at top of the model
310.4. Damping of the Outgoing Waves

310.4.1 Comparison of Rayleigh Damping and Caughey 4th Order Damping

As mentioned before, Caughey damping in general will damp out the motions at specified modes (frequencies) to be specified which could also be the natural frequencies of the system. Depending on the type of damping to be used, the response of those modes would be affected. In order to observe the damping effect on certain modes, a soil profile is made with thickness of 50 m and shear wave velocity of 100 m/s. For input motion, an Ormsby wavelet with frequency range of 0 to 7 Hz is considered at the base of model. The wave is propagated through the soil layer using the elastic transfer functions and comparison is made between the case which Rayleigh damping is used versus the case which the frequency independent damping is used in the model. The same procedure is done by using Caughey damping.

Figures (310.22) and (310.23) show the base motion (Ormsby wavelet), motion at the surface
Figure 310.16: Displacement time history of input motion (Ormsby Wavelet)

Figure 310.17: Comparison of displacement time histories of case 9, 10, and 11 at top of the model
Figure 310.18: Comparison of FFT of case 9, 10, and 11 at top of the model and input motion at the bottom of model.

Figure 310.19: Displacement time history of input motion (Ormsby Wavelet)
Figure 310.20: Comparison of displacement time histories of case 12, 13, and 14 at top of the model

Figure 310.21: Comparison of FFT of case 12, 13, and 14 at top of the model and input motion at the bottom of model
considering the frequency independent damping, motion at the surface using Rayleigh wave (frequency dependent), motion at the surface using Caughey damping of 4th order (frequency dependent) as well as how Rayleigh and Caughey damping ratio change with frequency. It can be observed how the response is affected at different modes using Rayleigh damping versus using Caughey damping.

![Graph](image.png)

Figure 310.22: Comparison of obtained motion at the surface using frequency independent damping and frequency dependent Rayleigh damping.

### 310.4.2 Parametric Study on Effect of Rayleigh Damping on Reflected Waves

As mentioned in previous chapter, one of the issues of the modeling in dynamic analysis is reflecting of the motions from the boundaries since there are limitations regarding the size of the problems we can model. In order to reduce the computational cost of the problems, the size of the mesh has to be reduced. By reducing the size of the model the chance of reflecting the motions from the boundaries gets higher since there is less volume for the waves to get dissipated.

There are different ways to reduce reflection of the waves from the numerical boundaries such as PML, viscous dampers, infinite elements, or considering Rayleigh damping for specific elements. Presented here show the results of wave propagation models considering Rayleigh damping. There are different damping patterns used here such as constant damping ratio for all the elements in the damping zone or linear pattern of increasing the damping ratio.

In order to find the Rayleigh damping coefficients, two frequencies have to be considered. In these examples both cases of using the natural frequencies of the soil column and also using the dominant
Figure 310.23: Comparison of obtained motion at the surface using frequency independent and frequency dependent Caughey 4th order damping.

periods of the motions are used and comparison is made. Different shear wave velocities and input motion frequencies are used which is mentioned for each case. The input motion considered for simulation is Ricker wavelet considering different dominant frequencies. $V_s$ is the soil profile shear wave velocity and $f_r$ is the frequency of the Ricker wavelet, and $x_i$ is the Rayleigh damping ratio at considered frequencies.

The height of the finite element model is 60m and boundary conditions are introduced in order to model 1C wave propagation. The motion is imposed at one side of the model and Rayleigh damping is applied to couple of the elements on the other side of the model in order to damp out the waves. Results are recorded at the boundary of damped and undamped zones.

Figure (310.24) shows the comparison of time histories for the soil column with shear wave velocity of 100m/s and input motion frequency of $8Hz$. Frequencies used to calculate Rayleigh damping coefficients in this case are natural frequencies of the soil column. The same damping ratio is used for all the damping zone elements. It can be observed that the one with constant damping ratio of 0.5 has done better job in terms of damping out the reflected motions.

Same analysis is done by using frequencies of $6Hz$ and $12Hz$ for the Rayleigh damping. As shown in Figure (310.25), in this case the reflected waves are damped out more comparing to previous case where natural frequencies of the soil were used for Rayleigh damping. This fact shows that the frequencies to be used for calculating the Rayleigh damping coefficients, do not have to be the natural frequencies of the soil which sometimes used in practice and depends on the frequency range of the input motion as
well. Since the Ricker wavelet used here has a peak frequency of $8Hz$, the higher values of frequencies should be used for Rayleigh damping coefficients. Figure (310.26) shows the results of the same case except that linear increasing pattern is used for damping ratio of the 5 elements in damping zone. It seems that using the linear pattern starting from 0.3 to 1.1 results in less reflected motions.

The reason could be because of the nature of Rayleigh damping which is frequency dependent. So different damping ratios are observed at different frequencies. In deed by changing the damping ratio at each element, five different patterns of Rayleigh damping are being used which has more capability of absorbing motions with different frequencies and amplitudes.

![Displacement time history considering Rayleigh damping using natural frequencies of the soil, $V_s=100m/s$, $f_r=8Hz$](image)

**Figure 310.24:** Displacement time history considering Rayleigh damping using natural frequencies of the soil, $V_s=100m/s$, $f_r=8Hz$

Figures (310.27) and (310.28) show the displacement time histories for same patterns of constant damping ratio for all elements in the damping zone and linearly increasing damping ratios respectively but for shear wave velocity of $V_s = 300m/s$ and input motion frequency of $5Hz$. Same conclusion can be made here as previous case regarding the pattern of damping ratios and frequencies to be used for Rayleigh damping.

Figures (310.29) and (310.30) are comparisons of recorded displacement time histories between patterns of same damping ratio for damping zone elements, damping ratio changes along the length of damping zone, and case of with out damping. Figure (310.29) is the case which shear wave velocity of the soil column is $100m/s$ and frequency of input Ricker motion is $1Hz$ while soil profile used in Figure (310.30) has shear wave velocity of $300m/s$ with input motion frequency of $8Hz$.

It can be observed that in case of having no physical damping, waves are getting trapped in the model and are reflecting back from the boundaries. Displacement time histories obtained from mentioned
patterns of damping ratios have minor differences which does not mean always will be this close but still the pattern of using linearly increasing of damping ratio seems to do a better job for damping the reflecting waves.

In order to have a better understanding of these patterns of damping, wave propagation through the depth of model is recorded for case of shear wave velocity of $100m/s$ and input motion frequency of $8Hz$.

Displacement time histories in Figures (310.31) to (310.33) show wave propagation through the model for cases of using uniform damping ratios, linearly increasing damping ratios, and with out damping.
respectively. By looking at the wave propagation through the whole soil profile it can be concluded that for this soil profile using the linearly increasing of damping ratios does a better job for damping the reflected motions at different depths.

Figure (310.34) shows the comparison of cumulative total energy time histories for the soil profile with shear wave velocity of 100 m/s and input motion frequency of 8 Hz for different Rayleigh damping patterns of uniform, increasing linearly, and case of no damping. What is expected to be observed is that total energy keeps increasing until the input motion gets to zero in time which energy should remain
constant unless there are waves reflecting back from boundaries. As it is shown, total energy slightly increase by time due to the reflected motions. This difference is much higher for case of no physical damping used since higher portion of the motions will get trapped in the model.

In order to be able to see the effect of size of damping zone on reflected motions, analysis is done on the soil profile with shear wave velocity of 100 m/s and frequency of 8 Hz for input motion. Comparison of displacement time histories for different size of the damping zones is shown in Figure (310.35). As expected, by reducing the size of damping zone, more waves are reflecting back from model boundaries.

The effect of number of elements to be used in damping zone is also studied here. Comparison is
Figure 310.31: Displacement time history in depth considering homogeneous damping (15m), $\xi=0.5$, $V_s=100\text{m/s}$, $f_r=8\text{Hz}$

made for cases which the size of the damping zone is the same but the size of the elements (and therefore number of the elements) in that zone is changed. The size of the damping zone assumed to be 15m while the number of the elements used in that zone is considered to be 3, 5, and 15. The comparison for this change of number of the elements is shown in Figure (310.36). Rayleigh damping ratio with pattern of increasing linearly from 0.3 to 1.1 is used. As it is observed, by reducing number of elements in the damping zone, the amount of reflected waves are getting higher.
Figure 310.32: Displacement time history in depth considering linear increasing of $\xi$ in Rayleigh damping (every 3m), $\xi=0.3, 0.5, 0.7, 0.9, 1.1$, $V_s=100\text{m/s}$, $f_r=8\text{Hz}$
Figure 310.33: Displacement time history in depth with out considering damping, $V_s = 100\text{m/s}$, $f_r = 8\text{Hz}$

Figure 310.34: Comparison of energy time history by considering different Rayleigh damping patterns (in the non-damping zone at the middle of model), $V_s = 100\text{m/s}$, $f_r = 8\text{Hz}$
Figure 310.35: Comparison of displacement time histories for different size of damping zones

Figure 310.36: Displacement time history at a point in the non-damping zone close to the boundary of imposing motion
310.5 Mesh Size Effects for Linear (8 Node Brick) and Quadratic (27 Node Brick) Finite Elements on Wave Propagation

Generally, the results of numerical analysis using finite element method technique for the dynamic problem are affected by size of mesh (grid spacing). According to Argyris and Mlejnek (1991), about 10 nodes per wavelengths are required to simulate accurately for the given frequency and fewer than 10 nodes may induce an artificial damping due to the numerical reason.

![Figure 310.37: One dimensional column test model to inspect the mesh size effect](image)

In this section, mesh size effect is inspected to decide an appropriate size of the mesh to build finite element model for verification. One dimensional column model is built as shown in figure 310.37. Total height of the model is 1000 m. Two models are built with element height of 20 m and 50 m, and each model have two different shear wave velocities (100 m/s and 1000 m/s). Density is set as 2000 kg/m$^3$, and Poisson’s ratio is set as 0.3, for all test models. Various cases are set and tested as shown in table 310.1. Both 8 node and 27 node brick elements are used for all models. Thus, total 24 parametric study cases are inspected. Linear elastic elements are used for all analyses. All analyses are performed in time domain with Newmark dynamic integrator without any numerical damping ($\gamma = 0.5$, and $\beta = 0.25$, no numerical damping, unconditionally stable).

Ormsby wavelet (Ryan, 1994) is used as an input motion and imposed at the bottom of the model.
Table 310.1: Analysis cases to determine a mesh size

<table>
<thead>
<tr>
<th>Case number</th>
<th>Vs (m/s)</th>
<th>Cutoff freq. (Hz)</th>
<th>Element height (m)</th>
<th>Max. propagation freq. (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>3</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>8</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>15</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>3</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>8</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>15</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>3</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>15</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>3</td>
<td>20</td>
<td>0.5</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>8</td>
<td>20</td>
<td>0.5</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>15</td>
<td>20</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Figure 310.38: Ormsby wavelet in time and frequency domain with flat frequency content from 5 Hz to 20 Hz

Ormsby wavelet features a controllable flat frequency content with formulation shown in equation 310.3.

\[
f(t) = A\left(\frac{\pi f_4^2}{f_4^2 - f_3^2} \text{sinc}(\pi f_4(t - t_s))^2 - \frac{\pi f_3^2}{f_1 - f_3} \text{sinc}(\pi f_3(t - t_s))^2\right) - \left(\frac{\pi f_2^2}{f_2 - f_1} \text{sinc}(\pi f_2(t - t_s))^2 - \frac{\pi f_1^2}{f_2 - f_1} \text{sinc}(\pi f_1(t - t_s))^2\right)
\]

(310.3)

where \(f_1\) and \(f_2\) define the lower range frequency band, \(f_3\) and \(f_4\) define the higher range frequency band, \(A\) is the amplitude of the function, and \(t_s\) is the time that maximum amplitude is happening, and
\[ \text{sinc}(x) = \frac{\sin(x)}{x}. \] Figure 310.38 shows an example of Ormsby wavelet with flat frequency content from 5 Hz to 20 Hz.

Figure 310.39: Number of nodes per wavelength along frequencies, and element sizes (a) \( V_s = 1000 \) m/s (b) \( V_s = 100 \) m/s

For this example, cutoff frequencies of Ormsby wavelets are set as 3, 8, and 15 Hz (table 310.1). Figure 310.39 shows number of nodes per wavelength along frequencies and figure 310.40 – 310.46 show comparison of analysis results. As shown in figure 310.40, case 1 and 7 (analysis using Ormsby wavelet with 3 Hz cutoff frequency) predict exactly identical results to the analytic solution in both time and frequency domain. Since, number of nodes per wavelength for both cases are over 10 (see figure 310.39(a) and table 310.1, all cases under 3 Hz shows more than 10 nodes per wavelength), those exact results are expected.

Increasing cutoff frequency from 3 Hz to 8 Hz induces numerical errors as shown in figure 310.41. In frequency domain, both 10 m and 20 m element height model with 27 node brick element predict exactly same results with the analytic one. However, in time domain, asymmetric shape of time history
Figure 310.40: Comparison between (a) case 1 (top, $V_s = 1000$ m/s, 3 Hz, element size = 10m) and (b) case 7 (bottom, $V_s = 1000$ m/s, 3 Hz, element size = 20m)
Figure 310.41: Comparison between (a) case 2 (top, Vs = 1000 m/s, 8 Hz, element size = 10m) and (b) case 8 (bottom, Vs = 1000 m/s, 8 Hz, element size = 20m)
Figure 310.42: Comparison between (a) case 3 (top, $V_s = 1000$ m/s, 15 Hz, element size = 10m) and (b) case 9 (bottom, $V_s = 1000$ m/s, 15 Hz, element size = 20m)
displacements are observed. Observations from top of 8 node brick element models show more numerical error in both time and frequency domain due to the decreasing number of nodes per wavelength (figure 310.39). Figure 310.42 shows analysis results with 15 Hz cutoff frequency. Results from 27 node brick elements are almost same in frequency domain but asymmetric shapes are also observed in time domain. Decreasing amplitudes in frequency domain along increasing frequencies are observed from 8 node brick element cases.

Figure 310.44 – 310.46 show results predicted from Vs = 100 m/s cases. Similar as Vs = 1000 m/s cases, decreasing amplitude along increasing frequencies are observed in all cases. One interesting observation is bumps in frequency domain which can be seen at natural frequencies (natural modes) of the elements (n th mode of elements, $f = (2n-1)V_s/4H$, 2.5 Hz, 5.0 Hz, and so on). This observation may mean that if certain condition is satisfied between modes and size of the element, it will behave like frequency contents filter. Figure 310.43 shows possible explanation of this observation. As in the case of figure 310.43, nodes (circle in the figure) cannot capture harmonic oscillation of the frequency since amplitude of the oscillation is always zero. As a result, the frequency contents at the frequency cannot be predicted by the analysis.

The results shown here are used as a reference to determine mesh size and frequency range of input motions for the verification.
Figure 310.44: Comparison between (a) case 4 (top, $V_s = 100$ m/s, 3 Hz, element size = 10m) and (b) case 10 (bottom, $V_s = 100$ m/s, 3 Hz, element size = 20m)
Figure 310.45: Comparison between (a) case 5 (top, $V_s = 100$ m/s, 8 Hz, element size = 10m) and (b) case 11 (bottom, $V_s = 100$ m/s, 8 Hz, element size = 20m)
Figure 310.46: Comparison between (a) case 6 (top, $V_s = 100$ m/s, 15 Hz, element size = 10m) and (b) case 12 (bottom, $V_s = 100$ m/s, 15 Hz, element size = 20m)
310.6 Verification of the Seismic Input (Domain Reduction Method) for 3C, Inclined Seismic Wave Fields

310.6.1 Inclined, 3C Seismic Waves in a Free Field

In this section verification of the 3C wave propagation problem using Domain Reduction Method will be studied. In order to do so, a finite element model with dimensions of $10000m \times 50m \times 5000m$ is considered. Two cases are studied here with the source of motion (fault) to be located at $(x = 3000m, y = 0, z = 3000m)$ and $(x = 3000m, y = 0, z = 3000m)$. Figures (310.47) and (310.48) show these two models respectively.

Figure 310.47: Domain to be analyzed for the 1st stage of DRM with fault located at an angle of $45^\circ$ with respect to the top middle point of the model

Figure 310.48: Domain to be analyzed for the 1st stage of DRM with fault located at an angle of $34^\circ$ with respect to the top middle point of the model
The size of the elements is chosen to be 50m in all directions for both cases in order to reduce the computational time. The soil parameters are: shear wave velocity of 700 m/s, density of 1800 kg/m$^3$, and Poisson’s ratio of 0.1. Analyses for the fault slip model are done by applying the motion at the nodes of one element. This is done in order to represent the wave propagation starting from the fault using Multiple Support Excitation. This is representing the first stage of analysis of DRM in which a big model including the fault is considered for free field case in order to obtain the required motions for DRM layer. For simulating the second stage of DRM, a smaller model with dimensions of 240 m x 5 m x 70 m is considered as shown in Figure (310.49). The size of the plastic bowl is 200 m x 5 m x 50 m. Size of the elements for this model is chosen to be 5 m.

Displacement and acceleration time histories of corresponding nodes of DRM layer are obtained by interpolating between the results obtained from the first model. These displacement and accelerations are used to calculate the effective forces as an input for DRM analysis. Input motions to be used here are Ricker wave, Morgan Hill, and Kocaeli earthquakes. The maximum allowable frequency to be propagated through this model can be calculated based on Equation (310.4):

$$\Delta h \leq \lambda/10 = \frac{V_s}{10 f_{\text{max}}}$$  \hspace{1cm} (310.4)

Based on the shear wave velocity of 700 m/s and element size of 50 m, maximum allowable frequency to be propagated through this model would be 1.4 Hz for the original model and based on element size of 5 m would be 14 Hz for the DRM model.
310.6.1.1 Ricker Wavelets

Figure (310.50) shows the displacement time history and FFT of Ricker wave of 2\textsuperscript{nd} order with dominant frequency of 1 Hz and maximum amplitude occurring at 1 second.

![Displacement and FFT of Ricker wave](image)

Figure 310.50: Displacement time history and FFT of Ricker wave with dominant frequency of 1 Hz

The first case to be studied here is the one with the fault source located at \((x = 3000 m, y = 0, z = 3000 m)\) which has the angle of 45° with respect to the top middle point of the model. Results to be discussed here are comparison of displacement and acceleration time histories at the top middle point of the model \((x = 5000 m, y = 0, z = 5000 m)\) between the fault slip and DRM models. Comparison of displacement time histories in X and Z directions are shown in Figure (310.51). As it can be observed, the results of DRM model matches perfectly with the ones obtained from the fault slip model.

Figure (310.53) is the displacement and acceleration time history of a point located outside of DRM layer in X direction \((x = 10 m, y = 0, z = 40 m)\). As mentioned before in definition of DRM, no motion should come out of the DRM layer in case of free field. As shown in these figures, displacement and acceleration time histories at this point are zero which verifies this fact.

The same motion is applied to the model with fault source located at \((x = 2000 m, y = 0, z = 3000 m)\) which has the angle of 34° with respect to the top middle point of the model. Displacement time histories of the top middle point show the perfect match between results obtained from fault slip model with the ones obtained from DRM mode.

As shown in Figure (310.55), the second motion to be used for analysis is Ricker wave with frequency of 0.5 Hz and maximum amplitude occurring at 3 seconds. Figure (310.56) shows the displacement time histories of X and Z directions for the same point as before \((x = 5000 m, y = 0, z = 5000 m)\). As it is shown, results of the fault slip and DRM model are the same which verifies the solution from DRM formulation for this motion as well.
The third motion to be used is Ricker wave with frequency of $2\,Hz$ and maximum amplitude happening at 1 second as shown in Figure (310.57). Comparison of displacement time histories between the fault slip and DRM model has been done and shown in Figure (310.58) along X and Z directions respectively. In this case, results do not match for the top middle point of the model. The main reason is due to the frequency of the motion. The maximum allowable frequency to be propagated in the fault slip model is $1.4\,Hz$ while it is $14\,Hz$ in DRM model. Dominant frequency of the Ricker wave as input motion
Figure 310.53: Displacement and acceleration time history for a point outside of DRM layer in (x) direction

Figure 310.54: Comparison of displacements for top middle point using Ricker wave ($f = 1 Hz$) as an input motion

is $2Hz$. Frequencies above the $1.4Hz$ can not be propagated in the fault slip model while they will propagate in the DRM model. this can change the characteristics of the motion propagating through the model and is the main reason of differences between the obtained results.
Figure 310.55: Displacement time history and FFT of Ricker wave with dominant frequency of 0.5 Hz.

Figure 310.56: Comparison of displacements for top middle point using Ricker wave (f = 0.5 Hz) as an input motion.
Figure 310.57: Displacement time history and FFT of Ricker wave with dominant frequency of $2Hz$

Figure 310.58: Comparison of displacements for top middle point using Ricker wave ($f = 2Hz$) as an input motion
310.6.2 Vertical (1C) Seismic Waves in a Free Field

310.6.2.1 Morgan Hill and Kocaeli Earthquakes

In order to investigate more, Morgan Hill and Kocaeli earthquakes are used as an input motions for the same models as before. These earthquakes were recorded during the ground shaking and obtained from PEER motion database. Figure (310.59) shows the acceleration time history and FFT of Morgan Hill earthquake with major frequency range of up to $4Hz$. Acceleration time history and FFT of Kocaeli earthquake are shown in Figure (310.60). Major part of the frequency range for Kocaeli earthquake is up to frequency of $4Hz$.

![Acceleration time history and FFT of Morgan Hill earthquake](image1)

![Acceleration time history and FFT of Kocaeli earthquake](image2)

Figure 310.59: Acceleration time history and FFT of Morgan Hill earthquake

Figure 310.60: Acceleration time history and FFT of Kocaeli earthquake

Figure (310.61) shows the displacement time histories of the top middle point of the model for
Morgan Hill earthquake while the ones from Kocaeli earthquake are shown in Figure (310.62). As it is observed, results of fault slip model and DRM model do not match since the majority of the energy in the earthquake is in the range of up to $4Hz$ which is higher than the maximum allowable frequency to be propagated in the original model ($1.4Hz$).

Figure 310.61: Comparison of displacements for top middle point using Morgan Hill earthquake as an input motion

Figure 310.62: Comparison of displacements for top middle point using Kocaeli earthquake as an input motion

In order to investigate more regarding the frequency content issue, Kocaeli acceleration time history is considered and frequencies above $1.4Hz$ are filtered out of the record. Acceleration time history and
FFT of the filtered record are shown in Figure (310.63). The majority of the energy is in the frequency range of below $1.4Hz$ while still there are frequencies up to $2Hz$ in the motion as can be observed in FFT of the filtered motion.

![Acceleration time history and FFT of filtered Kocaeli earthquake](image)

Figure 310.63: Acceleration time history and FFT of filtered Kocaeli earthquake

Figure (310.64) shows the displacement time histories for the same point as the one studied for the actual record. As it is observed, the obtained time histories match perfectly between the case of fault slip and DRM models. Figure (310.65) shows the acceleration time histories. Comparing the time histories shows an acceptable match between the results. There are tiny differences in acceleration time histories (specially at the peaks) which can be due to the fact that there are still frequencies above $1.4Hz$ in the input motion but with much less impact in terms of amplitude.
Figure 310.64: Comparison of displacements for top middle point using filtered Kocaeli earthquake as an input motion

Figure 310.65: Comparison of accelerations for top middle point using filtered Kocaeli earthquake as an input motion
310.6.3 Earthquake-Soil-Structure Interaction Verification for Simulated Northridge Seismic Motions

Figure 310.66: Finite element model to be used on analyses with input motions computed by integration equation (x-z plane view)

More realistic example is shown here. Seismic wave fields of Northridge earthquake simulated by program fk are applied as an input motion for this example. Figure 310.66 shows x-z plane view of three dimensional model. Similar as analytic case, using fk program, acceleration and displacement fields are generated at all nodes in DRM layer.

Figure 310.67 shows analysis results observed at the top-midpoint of the finite element model. As shown in figure 310.67, both results show perfect match.

310.6.4 Curious Case of 1C versus 3C modeling

To inspect more, artificial downhole array is prepared as shown in figure 310.68. Total 2 observation points are set on 0 m, and 50 m depth from the ground surface. one dimensional site response analyses are performed along artificial downhole array using DEEPSOIL v5.0 (Hashash and Park, 2002). 1D soil column model is built to run DEEPSOIL with identical soil properties to finite element model. Linear time domain site response analyses are performed. Displacements recorded at 200 m depth are used as an input motion. Site response analyses results on the observation points are compared with fk, and
Figure 310.67: Comparison between results computed from program fk and finite element analysis, observed at the top middle point of the finite element model.
finite element analyses results.

Figure 310.68: Comparison between analytic solution and FEM analysis result observed at top, middle point of the model (SV (imposed on x direction) Ricker wave input with 0°, x component)

Figure 310.69 – 310.74 are analyses results. Figure 310.69, 310.70, and 310.71 show comparison of results observed at the ground surface, EW, NS, and UD components, respectively. For the case of EW and NS components, 1C site response analyses results predict similar response as fk and FEM results compared to UD case. For all cases, 1C analyses results shows larger amplitude especially on UD case, 1Hz frequency contents show unrealistic response amplification. The same trend can be observed at 50 m depth cases (figure 310.72 – 310.74).

Possible explanation are as follows. fk results includes all components of waves (body and surface) and interaction between them. However, 1C wave propagation analyses cannot incorporate such effect. Also, 1C analyses is very sensitive to material properties (stiffness, damping ratio, and so on) and frequency contents of input waves.
Figure 310.69: Comparison between results computed from program fk, finite element analysis, and 1C analysis, observed at the top middle point, EW component
Figure 310.70: Comparison between results computed from program fk, finite element analysis, and 1D analysis, observed at the top middle point, NS component
Figure 310.71: Comparison between results computed from program f_k, finite element analysis, and 1C analysis, observed at the top middle point, UD component
Figure 310.72: Comparison between results computed from program fk, finite element analysis, and 1C analysis, observed at the depth = 50m, EW component
Figure 310.73: Comparison between results computed from program fk, finite element analysis, and 1C analysis, observed at the depth = 50m, NS component
Figure 310.74: Comparison between results computed from program fk, finite element analysis, and 1C analysis, observed at the depth = 50m, UD component
310.6.5 Earthquake-Soil-Structure Interaction for Surface and Embedded Structures


310.6.5.1 Uniform half-space

310.6.5.2 Layered half-space

310.6.5.3 Layered over rigid lower boundary
310.7 Case History: Simple Structure on Nonlinear Soil

310.7.1 Simplified Models for Verification
Due to the complexity of full scale finite element models it is helpful to perform preliminary tests on simplified models in order to verify the adequacy of the time and mesh discretization with respect to the input motion. It also provides good insight in the performance of the nonlinear material model. To achieve this a series of tests on a one-dimensional soil column have been proposed:

- Static pushover test on nonlinear soil column
  Through the static pushover test the behavior of the nonlinear material model can be verified.

- Dynamic test of elastic soil column
  By applying an earthquake motion to the elastic soil column it can be tested whether the selected grid spacing is capable of representing the motion correctly without filtering out any relevant frequencies. This test also allows to choose appropriate damping parameters. It should be noted that this is additional (small) damping that is used for stability of the numerical scheme and should not be relied upon to provide major energy dissipation. Major energy dissipation should be coming from inelastic deformations of the SFS system.

- Dynamic test of nonlinear soil column
  Finally the stability and the accuracy of the numerical method can be examined by applying the earthquake motion to the nonlinear column of soil. A second analysis with a time step reduced by 50% should not give a significantly different result.
  Furthermore it will be examined how propagation through an elastic-plastic material will change the frequency content of the motion.

310.7.1.1 Model Description
The one-dimensional soil column used for verification has the same depth and element sizes as the 2d and 3d models that will be addressed later. Its total depth is 10.5 meters and it consists of a single stack of 8-node brick elements of 1.5 meters side length. In order to achieve one-dimensional wave propagation in vertical direction the movement of four nodes at each level of depth is constrained to be equal. The input motion is applied to the four nodes at the base of the model. As input motions four time histories from the Northridge Earthquake are selected (Figure 310.75).

The material properties of the soil are given in Table 310.7.1.1.
Figure 310.75: Acceleration time histories and Fourier amplitude spectra’s of the selected ground motions

<table>
<thead>
<tr>
<th>Friction angle $\phi'$</th>
<th>37°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undrained shear strength $c_u$</td>
<td>10 kPa</td>
</tr>
<tr>
<td>Mass density $\rho$</td>
<td>1800 kg/m$^3$</td>
</tr>
<tr>
<td>Shear wave velocity $v_s$</td>
<td>200 m/s</td>
</tr>
</tbody>
</table>

The discretization parameters, the time step $\Delta t$ and the maximum grid spacing $\Delta h$, are determined following the guidelines outlined in Section 502.3.1. This yields a maximum grid spacing of

$$\Delta h \leq \frac{v_s}{10f_{max}} = \frac{200}{10 \times 10} = 2 \text{ m} \quad (310.5)$$

For the following analysis $\Delta h = 1.5 \text{ m}$ is selected. The maximum time step is

$$\Delta t \leq \frac{\Delta h}{v_s} = \frac{1.5}{200} = 0.0075 \text{ s} \quad (310.6)$$

Taking into account a further reduction of the time step by about 60% due to the use of nonlinear material models $\Delta t = 0.002 \text{ s}$ is chosen.
310.7.1.2 Static Pushover Test on Elastic-Plastic Soil Column

For the static pushover test, an elastic perfectly plastic Drucker-Prager material model as specified in Table 310.7.1.1 is used.

After applying self weight a horizontal load of 100 kN is applied to a surface node in increments of 0.1 kN. The system of equations is solved using a full Newton-Raphson algorithm. The predicted shear strength of the first element that is expected to fail, the one at the surface, is:

\[
\tau_f = c_u + z \rho g \tan \phi' = 10 + 0.75 x 1.8 x 9.81 \tan 37^\circ = 19.98 \text{kPa} \tag{310.7}
\]

where \( z \) is the depth of the center of the first element.

Self weight produces the following stresses in the element at the surface:

\[
\sigma_x = \sigma_y = 8.83 \text{kPa} \\
\sigma_z = 13.24 \text{kPa}
\]

The maximum shear stress is

\[
\tau_{\text{max}} = \sqrt{\left(\frac{\sigma_z - \sigma_x}{2}\right)^2 + \tau_{xz}^2} \tag{310.8}
\]

The theoretical failure load can be obtained as follows:

\[
P_f = \tau_{xz} A = \sqrt{\tau_f^2 - \left(\frac{\sigma_z - \sigma_x}{2}\right)^2} A = 44.7 \text{kN} \tag{310.9}
\]

The static failure load is underestimated by about 6%. This accuracy is acceptable for the given model because the boundary conditions cannot assure constant stresses at a given depth (no shear stress is applied to the lateral surfaces).

310.7.1.3 Dynamic Test on Elastic Soil Column

In order to test the spatial discretization of the model an earthquake motion is propagated through an elastic soil column. The grid spacing of the finite element mesh can be considered sufficiently fine if frequencies up to \( f_{\text{max}} = 10 \) Hz are represented accurately in the numerical analysis. A good way to verify this is to calculate transfer functions between the base and the surface of the soil column.
Because transfer functions don’t depend on the input motion they can easily be compared with closed form solutions.

The transfer function of a soil deposit describes the amplification between the frequencies of the motion at the base and at the soil surface:

$$TF(\omega) = \frac{u(z = 0, \omega)}{u(z = H, \omega)}$$  \hspace{1cm} (310.10)

where $z$ is the depth measured from the surface and $H$ is the thickness of the soil deposit above the bedrock. $\omega = 2 \pi f$ is the circular frequency.

For elastic soil with viscous damping the wave equation can be written as (Kramer, 1996a)

$$\rho \frac{\partial^2 u}{\partial t^2} = G \frac{\partial^2 u}{\partial z^2} + \eta \frac{\partial^3 u}{\partial z^2 \partial t}$$ \hspace{1cm} (310.11)

$\eta$ is the damping coefficient, defined as

$$\eta = \frac{2G}{\omega} \xi$$ \hspace{1cm} (310.12)

where $\xi$ is the frequency independent hysteretic material damping.

After solving the wave equation the transfer function can be written as

$$TF(\omega) = \frac{1}{\cos \omega H/v_s^*}$$ \hspace{1cm} (310.13)

where $v_s^*$ is the complex shear wave velocity

$$v_s^* = \sqrt{\frac{G^*}{\rho}} = \sqrt{\frac{G(1 + i2\xi)}{\rho}}$$ \hspace{1cm} (310.14)

In a finite element model with mass- and stiffness proportional Rayleigh damping the damping coefficient $\eta$ is constant. Therefore the hysteretic material damping ratio $\xi$ needs to be frequency dependent in order to satisfy Equation 310.12. Solving Equation 310.12 for $\xi$ and substituting it into Equation 310.14 and then into Equation 310.13 yields a new transfer function:

$$TF(\omega) = \frac{1}{\cos \left(\omega H \sqrt{\frac{\rho}{G + i\omega \eta}}\right)}$$ \hspace{1cm} (310.15)

Figure 310.76 shows a comparison between the closed form solution and the numerical transfer functions obtained from the finite element analysis. Rayleigh damping is used to obtain the damping matrix $C$:

$$C = \alpha M + \beta K$$ \hspace{1cm} (310.16)

The analysis are performed using stiffness proportional Rayleigh damping of $\beta = 0.001$ and $\beta = 0.01$. No mass proportional damping is applied ($\alpha = 0$). The damping coefficients of the closed form solution are chosen to be $\eta = \beta G$.
It can be seen that the numerical transfer functions are very close to the closed form solutions for $\eta = \beta G$. The peak corresponding to the second natural frequency of the soil layer is slightly shifted to the right in the result of the FE analysis. For the FE analysis the Rayleigh damping cannot be reduced any further as the solution would become unstable. This result proves that a FE analysis involving Rayleigh damping with $\alpha = 0$ and $\beta = \eta / G$ is equivalent to the closed form solution of the wave equation with frequency-dependent hysteretic material damping.

Based on the above observations a stiffness proportional Rayleigh damping of $\beta = 0.01$ is selected for the finite element analysis. This choice damps frequencies above 10 Hz appropriately.

### 310.7.1.4 Dynamic Test on Elastic-Plastic Soil Column

As the next step an elastic-plastic material model of Drucker-Prager type with kinematic strain hardening has been selected. Previous analysis involving material with isotropic hardening have proved to be unsuitable because energy can only be dissipated as the yield surface expands. For dynamic problems this can lead to an unreasonably large extension of the yield surface, especially if resonance frequencies are present. Therefore only kinematic hardening has been selected in this analysis.

The analysis were performed with four different ground motions using time steps of $\Delta t = 0.002s$ and
\[ \Delta t = 0.001s. \] A linear integrator without iterations within a time step was used. All ground motions were scaled to a maximum acceleration of 1g. For comparison the analysis were also performed on elastic material. Figure 310.77 shows the displacement time histories at the surface for all four ground motions. While the overall shapes of the displacements are the same as for the elastic case there is some residual plastic displacement resulting in the time histories of the Century City motions.

The Fourier amplitude spectra’s of the acceleration recorded at the surface (Figure 310.78) have the same general shape for the case of elastic and elastic-plastic material. The amplification at the first resonance frequency \( f = 4.75Hz \) is bigger in the elastic analysis. Higher frequencies resulting from plastic slip are damped out effectively in the nonlinear analysis.

Figure 310.79 shows the acceleration time history at the upper node of the lowest element, that is the first free node above the base. The record shows large peaks of the order of about 6 g. These peaks are caused by plastic slip and counter balancing of the resulting plastic deformation. The periods of the peaks are of the order of a few time steps, they add a very high frequency component to the
acceleration. Because these frequencies are due to a purely numerical phenomenon, they should not be allowed to propagate through the model. This can be achieved easily by specifying an appropriate numerical procedure (Newmark with appropriate combination of $\gamma$ and $\beta$) or with Rayleigh Damping.

As for the elastic model transfer functions were also computed for the nonlinear model. In Figure 310.80 the transfer functions between the acceleration at the soil surface and the base are compared. The functions for the nonlinear model are not smooth anymore but the general shape is the same as for the linear elastic model, i.e. the first natural frequency of the layer is clearly visible. The peaks that are present in the range of 25 Hz are purely numerical as they appear due to the division by a very small value.

A second set of analysis performed with half the time step of the previous analysis gives an idea of the accuracy of the numerical method. In Figure 310.81 the difference between the displacement (or acceleration) of the analysis with $\Delta t = 0.002s$ and $\Delta t = 0.001s$, divided by the corresponding maximum value is given for the entire time history:

$$\Delta d = \frac{d_{0.002}(t) - d_{0.001}(t)}{\max d_{0.001}} \quad \text{and} \quad \Delta a = \frac{a_{0.002}(t) - a_{0.001}(t)}{\max a_{0.001}} \quad (310.17)$$
In Figure 310.82 an integral measure for the difference in displacements and accelerations between the two analysis is given for all depths. The integral measures are defined as

\[
\text{diff}_d = \frac{1}{\max|d|} \frac{1}{T} \sum_{0}^{T} |d_{0.002}(t) - d_{0.001}(t)| \, dt
\]  

(310.18)

\[
\text{diff}_a = \frac{1}{\max|a|} \frac{1}{T} \sum_{0}^{T} |a_{0.002}(t) - a_{0.001}(t)| \, dt
\]  

(310.19)

The integral differences in accelerations are quite large in the elements that are close to the base, that is where the motion is applied. Toward the surface the difference becomes smaller than 1%. This is a result of the fact that most of the plastic deformation occurs near the base which represents an undesired boundary effect. Again this result underlines the importance of an appropriate choice of the size of the computational domain.

With a point wise difference not exceeding 5% for accelerations and 2% for displacements the time step \( \Delta t = 0.002 \) s is sufficiently small to ensure stable and accurate results.
Figure 310.80: Transfer functions between acceleration at the soil surface and the base

### 310.7.1.5 2d Model

A 2d-model is proposed as a simplification of the full 3d-model. Representing a cross section of the full model it is expected to provide insight into its dynamic behavior while requiring considerably less computational resources. The 2d-model consists of one slice of eight-node brick elements as shown in Figure 310.83. The nodes of the two lateral faces are constrained to move together in x- and z-direction, the out-of-plane displacement in y-direction is fixed. The model approximates a plane strain situation.

The earthquake motion is applied to the model by the DRM method.

### 310.7.1.6 Input Motions

As input for the 2d model the motion from the Northridge earthquake recorded at LA University Hotel (Figure 310.75) is used. The acceleration time history is scaled to a peak ground acceleration of 1 g. Motion is applied in x-direction only, that is, this is a 1–D wave propagation.

Acceleration time histories at all the nodes of the boundary layer are obtained by vertically propagating
Figure 310.81: Difference between results of analysis with different time steps, in percent of the maximum value

a plane wave using the program SHAKE91 (Idriss and Sun, 1992). Because the free-field model has to match the properties of the free field as represented by the finite element model for the reduced domain, only linear elastic material without strain dependent reduction of shear modulus and a constant amount of hysteretic material damping is used in the SHAKE91-analysis. The earthquake motion obtained in this way corresponds to a shear wave propagating upward through a homogeneous linearly elastic half space.

The acceleration time histories from the SHAKE91-analysis are then integrated twice to obtain displacements. Before integration the acceleration and velocity time histories are transformed into Fourier space, multiplied with a high pass filter and transformed back into time domain. Then a simple parabolic baseline correction is performed in order to obtain zero initial, final and mean values.
Figure 310.82: Averaged differences between results of analysis with different time steps

Figure 310.83: Two-dimensional quasi-plane-strain model
310.7.1.7 Boundary Conditions

Different boundary conditions are tested on the free-field model. First all outside boundary nodes are fully fixed as shown in Figure 310.84 a). Then they are released and attached to dash pots that are both perpendicular and tangential to the boundary (schematically shown in Figure 310.84 b)). The dash pots perpendicular to the boundary are specified to absorb p-waves, those tangential to the boundary to absorb s-waves. Because in this configuration no displacement constraint is imposed to the model on the faces at $x = \pm 10.5$ meter the horizontal at-rest soil pressure has to be applied to the corresponding nodes manually. This is done by recording the reaction forces in the model with fixed boundaries and applying them with opposite sign to the model with absorbing boundaries. The horizontal displacements after applying self weight should be very small.

Figure 310.85 shows results from a free-field analysis on a homogeneous elastic model. Displacements on an exterior boundary node as well as transfer functions between a point at the surface and a point on the exterior boundary of the plastic bowl are presented for the two configurations of boundary conditions shown in Figure 310.84. It can be seen that the displacements outside the plastic bowl in the model using absorbing boundary conditions are much larger compared to the model with fixed boundaries. This result is as expected considering the immediate proximity of the boundary. It also gives an idea about the constraints the fixed boundary imposes on the motions. The transfer function in Figure 310.85 b) is defined as the ratio between the Fourier amplitude spectra of a point at the surface and a point on the exterior boundary:

$$TF(\omega) = \frac{A_1(\omega)}{A_2(\omega)}$$ (310.21)
where $A_1(\omega)$ is the Fourier amplitude spectrum of the acceleration time history at the point $(x,z) = (0,0)$ m and $A_2(\omega)$ the corresponding spectrum at the point $(x,z) = (9.0,-7.5)$ m. The figure shows that the large peak representing the first natural frequency of the system, corresponding to a standing shear wave in a soil layer of 10.5 meter depth, gets reduced considerably by the absorbing boundary. An energy build-up in the model due to reflection of waves on the model boundaries can be reduced effectively with the configuration of boundary conditions shown in Figure 310.84 b). By releasing the fixed node at $(x,z) = (0,-10.5)$ m the resonance peak could be reduced by another 10% approximately, however at the cost of remaining permanent displacements at the end of the analysis.

Alternatively to imposing a rigid constraint to a single node at the base the model can be prevented to move horizontally as a rigid body through uniaxial springs. This gives the possibility to adjust the frequency of the eigenmode that corresponds to a vertically propagating plane shear wave. By appropriately choosing the spring constants the model can therefore be adjusted in such a way that it represents the natural frequency of a soil deposit on bedrock.

### 310.7.1.8 Structure

Four very simple structures are chosen to illustrate the effects of dynamic SFSI. A beam-column element of length $L$ and moment of inertia $I_y$ is fixed to a footing. A lumped mass of $M = 100,000$ kg is
added to the translational degrees of freedom of the top of the structure. The footing is 0.5 m deep, spans over four soil elements and is rigidly connected to the adjacent soil nodes. Its Young’s modulus is chosen large enough so that the footing can be considered rigid. The mass density of the footing is \( \rho = 2400 \text{ kg/m}^3 \), the column is considered massless. The moment connection between the nodes of the footing, having 3 (translational) degrees of freedom, and the 6 degrees of freedom of the nodes of the column is assured by a very stiff beam element that is connected to a node at the bottom and a node at the top of the footing. The column is then simply connected to the upper node of this auxiliary beam element.

The parameters of the four columns are chosen such that the second natural frequency, that is the natural frequency attributed to bending of the column (Figure 310.87 b)), is evenly distributed over the frequency range of the input motion (Figure 310.88). Structure 4 is designed such that it’s second natural frequency matches the largest spike in the input motion. Table 310.2 lists the properties of
Figure 310.88: Fourier amplitude spectrum of input motion with second natural frequencies of the 4 SFSI-systems

the structures used in the analysis. For the nonlinear columns a strain hardening material is chosen that consists of an initial elastic branch with tangent modulus $E$ and a post-yield branch with tangent modulus $0.2E$. The Young’s modulus for all four structures is $E = 210\, \text{GPa}$. The yield stress $f_y$ for structures 1, 2 and 4 is 20 MPa and for structure 3 it is 2 MPa.

### Table 310.2: Properties of the analyzed structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Length [m]</th>
<th>Stiffness $EI_y$ [MN m$^2$]</th>
<th>Mass [kg]</th>
<th>Yield Moment [kNm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.5</td>
<td>1680</td>
<td>100,000</td>
<td>800</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
<td>5670</td>
<td>100,000</td>
<td>1,800</td>
</tr>
<tr>
<td>3</td>
<td>2.5</td>
<td>13440</td>
<td>100,000</td>
<td>320</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>5670</td>
<td>100,000</td>
<td>1,800</td>
</tr>
</tbody>
</table>

310.7.1.9 Structure with Fixed Base

To begin with a parametric study of a series of structures with varying stiffnesses is analyzed. The stiffness is varied by changing the width of the column section. The different structures are expected to respond specifically to the frequency range of the input motion that is in the neighborhood of the natural frequency of the column. The input motion that is applied at the base of the structure has been recorded in a previous free-field analysis of the 2d-model.

The results of this parametric study are shown in Figures 310.89 and 310.90 for linear and nonlinear structures, respectively. The Fourier amplitudes spectra’s of the acceleration at the top of the structure
Figure 310.89: Parametric study of 15 linear structures with varying natural frequency.
Figure 310.90: Parametric study of 15 nonlinear structures with varying natural frequency
are plotted for 15 structures with variable natural frequency $f_n$. A line of equal frequency is also provided. The input motion is plotted in the background of the figure. It can be seen that the maxima of the frequency spectra’s are almost perfectly aligned along the line of equal frequency. This is even more obvious in the case of a linear structure. In that case the responses of the structures are very narrow banded. As the structure remains elastic the top of the structure oscillates mainly in its initial natural frequency. Lower and higher frequencies are eliminated to a great extent.

In the case of the inelastic structure there is clearly more damping and reduction of the responses at some (most) frequencies. The nonlinearity in structure is producing a longer effective period for the structure, and that effective period changes during shaking. This in turn widens the frequency range of structural response. That is, the response is lower, but the frequency characteristic is (much) wider.

A series of fixed-base analysis is also performed on the four structures mentioned in Section 310.7.1.8. The first natural frequencies of the four structures with its base fixed, corresponding to the second mode of vibration of the SFSI-model, are given in Table 310.3. It can be seen that the influence of the soil on the natural frequency of the SFSI-system increases as the overall stiffness of the structure increases.

<table>
<thead>
<tr>
<th>Structure</th>
<th>1\textsuperscript{st} natural frequency of fixed-base system [Hz]</th>
<th>2\textsuperscript{nd} natural frequency of SFSI-system [Hz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.71</td>
<td>2.07</td>
</tr>
<tr>
<td>2</td>
<td>9.82</td>
<td>3.89</td>
</tr>
<tr>
<td>3</td>
<td>25.1</td>
<td>5.53</td>
</tr>
<tr>
<td>4</td>
<td>5.75</td>
<td>3.52</td>
</tr>
</tbody>
</table>

Table 310.3: Eigenfrequencies of the analyzed models

### 310.7.1.10 Results

The results of the SFSI- as well as the fixed-base analysis are presented in the following. The displacements at the top of the nonlinear structures are recorded and plotted in Figure 310.91. It can be seen that the results from the SFSI- and the fixed-base-model differ considerably in terms of maximum as well as permanent displacement. In contrast to this the displacements at the base of the column are almost identical for the two models (results not plotted). Figure 310.92 displays the displacements at the top of structures 1 and 2 for all the combinations of linear and nonlinear soil and structures that have been analyzed. Due to the low yield moment the permanent displacement for structure 1 is relatively large in the analysis involving nonlinear columns. The results involving nonlinear columns on linear and on nonlinear soil are very similar in their overall shape, however permanent deformations are very different.
Figure 310.91: Displacements in x-direction at the top of the nonlinear structures

It seems that the forces that trigger plastic deformations in the column strongly depend on the behavior of the soil beneath the foundation.

In order to investigate the forces causing plastic deformations in the structures we look at the base moments between foundation and column. In Figure 310.93 the moments at the base of the linear structures are plotted.

For structures 1 and 4 the moments for the fixed-base model are higher than for the SFSI-model. This means that in this case neglecting the effects of SFSI leads to a conservative design. Structures 2 and 3 however have to resist higher moments when SFSI is taken into account. Because the SFSI-system is more flexible than the fixed-base structure its modes of vibration are excited by a different range of frequencies contained in the input motion. For a particular motion this can lead to resonance of the SFSI-system. This result is in contradiction with current engineering practice suggesting that neglecting SFSI in general leads to a more conservative design.

Figure 310.94 shows the moments at the base of structures 1 to 4, this time for the analysis involving nonlinear column elements. The evolution of the second natural frequency of the SFSI-system is also
Figure 310.92: Displacements in x-direction at the top of structures 1 and 2

provided as a qualitative indication for when plastic deformations occur. The base moments for structures 1 and 3 in the fixed-base- and the SFSI-analysis are very similar. Due to the low yield moment of the structure no resonance with the input motion occurs as a lot of energy is dissipated through plastic deformation.

Figure 310.95 shows an interesting aspect of nonlinear SFSI. In the analysis involving elastic-plastic soil the Fourier amplitudes of the moment at the base of the structure are reduced in the neighborhood of the natural frequency of the system. This is most likely due to dissipation of energy caused by elastic-plastic deformations in the soil that, in their turn, are a result of large loads provoked by resonance between the SFSI-system and the input motion.

As a measure of the plastic strain occurring beneath the footing the equivalent plastic strains averaged over all the Gauss points are calculated. The results are given at $t = 12\, \text{s}$ and at $t = 14\, \text{s}$, that is shortly before and after the largest plastic deformation occurs (Figures 310.96 and 310.97).

Plastic strains are larger in the analysis involving an elastic structure. This reflects the fact that elastic structures don’t dissipate any energy by themselves. For structure 2 no significant difference can be observed because of its high yield moment. Structure 4 is characterized by the same yield moment, its slightly smaller natural frequency however causes resonance with the input motion which leads to larger plastic strains beneath the footing. The largest plastic strains develop in the layer of elements adjacent to the boundary layer. This can be due to an input motion that isn’t fully compatible with the
elastic properties of the DRM-model. It should be possible to reduce these undesired plastic strains by either increasing the size of the soil model or by selecting a method to obtain the free-field motions that represents the soil properties of the DRM-model more closely.

310.7.2 Full nonlinear 3d Model

The 2d SFSI-model presented in the previous section is extended to a 3d model in the following. The goal is to show that the considerations for accuracy and stability of the numerical method obtained from the 1d-model remain valid for the 3d-model. Even if the simplicity of the analyzed problem doesn’t necessarily justify the additional computational effort it is important to show that it is possible to obtain
Figure 310.94: Moments at the base of the *nonlinear* column

Figure 310.95: Fourier amplitude spectra of moments at the base of *nonlinear* column, SFSI-analysis
reliable results for a problem that involves the following elements:

- 3d model with about 700 elements, 960 nodes and 2700 equations
- Elastic-plastic soil (Drucker-Prager with kinematic hardening)
- Nonlinear structure (bilinear material model)
- Ground motion applied through the Domain Reduction Method (DRM)
- Absorbing boundary of Lysmer type

310.7.2.1 Description of Model

The 3d model is based on the 2d model shown in Figure 310.86. In y-direction 6 more slices of $7 \times 14$ elements are added (Figure 310.98). The x-z plane at $y = 0$ represents a plane of symmetry. Lysmer boundaries are attached to all outside boundaries with the exception of the plane of symmetry and the
soil surface. The main difference to the 2d model is that 3d wave propagation is possible which leads to higher radiation damping.

The structure was chosen to have the same geometric and material properties as Structure 4 in the previous section.

310.7.2.2 Results

Some results of the 3d-analysis together with the corresponding data of the 2d analysis are presented in Figure 310.99. Due to limited memory only the first 20 seconds of the time history were processed. A more efficient implementation of the application of effective forces for the DRM-method inside the finite element code should solve this problem. The analysis took 66 hours to finish.

The displacements obtained at the top of the structure as well as the moments at its base are very close to the results of the 2d-analysis. This shows that the analysis provides reliable results for a full 3d nonlinear SFSI problem. The amplitude of the base moment is at several instances larger for the 3d-model than for the 2d-model. This can be explained with the fact that more energy is present in the 3d-model whereas the energy the structure can absorb is the same as in the 2d-model. Also it is obvious that the natural frequencies of the 3d-model are not exactly the same as for the 2d-model and therefore changes the dynamic behavior in a way that is almost impossible to predict beforehand.

Because of the simple geometry of the problem the 2d-model is absolutely sufficient for analyzing the forces acting on the structure. If one is interested in the stress history in the soil surrounding the footing then the 3d-model can provide valuable additional information.

...
Figure 310.99: Top: Displacements at the top of Structure 4, Bottom: Moments at the base of Structure 4
310.8 Lotung Large Scale Seismic Test (LLSST) Earthquake 07

310.8.1 Introduction

Figure 310.100 shows the G/Gmax and Damping data used for Pisano model.

![Figure 310.100: G/Gmax and Damping Curves](image)

310.8.2 Input motion and input method

We use the data from Lotung Large Scale Seismic Test (LLSST) which is operated by TaiPower and IES during the period from 1985 to 1990. We choose event 07 to verify our modeling. Seismic motion data is available for download [HERE].

310.8.3 Results
Figure 310.101: Time history comparison at different depths
Figure 310.102: Fourier spectrum comparison at different depths
Figure 310.103: Response spectrum comparison at different depths (damping ratio 5%)
Chapter 311

Verification and Validation for Static and Dynamic Behavior of Soil-Structure-Interaction

(In collaboration with Dr. Nima Tafazzoli, Dr. Yuan Feng, Dr. Han Yang, and Dr. Hexiang Wang)
311.1 Chapter Summary and Highlights
311.2 Solid-Beam Model-Comparison of Real-ESSI eigen frequencies with ANSYS and Sofistik

Figure 311.1 and Table 311.1

![Finite element model](image)

Figure 311.1: Finite element model

Table 311.1: Comparison of eigen frequencies obtained from Real-ESSI, ANSYS, and Sofistik

<table>
<thead>
<tr>
<th>Mode</th>
<th>Real-ESSI (Hz)</th>
<th>ANSYS (Hz)</th>
<th>Sofistik (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.4887</td>
<td>5.3868</td>
<td>5.439</td>
</tr>
<tr>
<td>2</td>
<td>7.1729</td>
<td>7.0711</td>
<td>7.311</td>
</tr>
<tr>
<td>3</td>
<td>12.6907</td>
<td>12.4670</td>
<td>12.751</td>
</tr>
<tr>
<td>4</td>
<td>13.197</td>
<td>13.1137</td>
<td>15.688</td>
</tr>
</tbody>
</table>
311.3 Solid-Beam Model-Comparison of model responses using elastic beams with 12dofs and 9dofs

311.3.1 FEM Model

Figure 311.2

![Finite element model]

Figure 311.2: Finite element model

311.3.2 Static Analysis

Figure 311.3

311.3.3 Dynamic Analysis-Applying Force

Figures 311.4 and 311.5

311.3.4 Dynamic Analysis-Applying Displacement

Figures 311.6 and 311.7 and 311.8
Figure 311.3: Comparison of static displacements (pseudo time) of the top nodes, Force time history applied to the top node

Figure 311.4: Comparison of displacement time histories of top node, Force time history applied to the top node, mass comes from the lumped mass added to the top node as well as the beams density
Figure 311.5: Comparison of displacement time histories of top node. Force time history applied to the top node, mass comes only from the beams density.

Figure 311.6: Comparison of displacement time histories of top node. Displacement time history applied at the bottom, mass comes only from the lumped mass added to the top.
Figure 311.7: Comparison of displacement time histories of top node. Displacement time history applied at the bottom, mass comes from the lumped mass added to the top node as well as the beams density.

Figure 311.8: Comparison of displacement time histories of top node. Displacement time history applied at the bottom, mass comes only from the beams density.
311.3.5 Comparison of eigen frequencies between models of using 9dof beam and 12dof beam

Table 311.2

<table>
<thead>
<tr>
<th>Mode</th>
<th>9dof</th>
<th>12 dof</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.253</td>
<td>5.412</td>
</tr>
<tr>
<td>2</td>
<td>6.867</td>
<td>7.112</td>
</tr>
<tr>
<td>3</td>
<td>9.399</td>
<td>12.570</td>
</tr>
<tr>
<td>4</td>
<td>11.60</td>
<td>13.183</td>
</tr>
<tr>
<td>5</td>
<td>11.76</td>
<td>13.479</td>
</tr>
<tr>
<td>6</td>
<td>12.95</td>
<td>14.605</td>
</tr>
<tr>
<td>7</td>
<td>15.97</td>
<td>15.998</td>
</tr>
<tr>
<td>8</td>
<td>18.85</td>
<td>19.101</td>
</tr>
<tr>
<td>9</td>
<td>25.27</td>
<td>21.378</td>
</tr>
</tbody>
</table>

311.3.6 Eigen modes of model using 12dof beam

Figures 311.9 and 311.10 and 311.11 and 311.12 and 311.13

311.3.7 Eigen Modes of model using 9dof beam

Figures 311.14 and 311.15 and 311.16 and 311.17 and 311.18

311.4 Validation Using UNR Soil Box Test Setup
Figure 311.9: Mode 1

Figure 311.10: Mode 2
Figure 311.11: Mode 3

Figure 311.12: Mode 4
Figure 311.13: Mode 5

Figure 311.14: Mode 1
Figure 311.15: Mode 2

Figure 311.16: Mode 3
Figure 311.17: Mode 4

Figure 311.18: Mode 5
Chapter 312

Verification and Validation for Dynamic Solid-Fluid Interaction

(In collaboration with Dr. Hexiang Wang)
312.1 Chapter Summary and Highlights

312.2 V&V Examples

312.2.1 Box sloshing

A numerical example of earthquake-driven box sloshing is provided here. The initial configuration is shown in figure 312.1. The length of the box is 30 meters and the height is 10 meters. The thickness is 3 meters filled with 8-meter-deep water. The box is shaken by a uniform 1D horizontal excitation plotted in figure 312.2. The box is modeled with elastic material with $E = 12\, GPa, \mu = 0.2$.

![Model configuration of 2D sloshing box](image)

The simulation result can be seen in figure 312.3. Clearly, elevations of free water surface can be observed under the excitation.

Another sloshing box example is driven by falling water flow. Figure 312.4(a) shows the result where solid domain and fluid domain has totally compatible mesh size (mesh size ratio 1:1). As mentioned before, VOF method has high requirement for the mesh size. In order to get accurate enough result with limited computation resources, refined mesh in the fluid domain is usually required for soil-structure-fluid interaction analysis. Analysis of model with discontinuous mesh is supported here through generalized interpolation scheme. Figure 312.4(b) shows the result of refined finite volume mesh in fluid domain (mesh size ratio 1:3).
Figure 312.2: Time history of 1D excitation

Figure 312.3: Simulation result of 2D sloshing box

Figure 312.4: Box sloshing under falling water
312.2.2 Dam Break

312.3 Verification & Validation

Verification and validation is crucially important to guarantee the accuracy of simulation result. Verification and validation (V & V) procedure of SFI problem involve three aspects: V & V for response of solid domain (i.e. V & V for RealESSI as a solver for solid mechanics), V & V for response of fluid domain (i.e. V & V for OpenFOAM as a solver for free surface flow) and V & V for SFI (i.e. V & V for interaction between solid domain and fluid domain).

Since RealESSI developed by Jeremić et al. (1988-2021) has rigorous V & V procedure, the focus here is on V & V of OpenFOAM and SFI.

312.3.1 Free Surface Flow validation

The functionality of OpenFOAM as a solver for free surface flow is validated in this section. A numerical validation test is conducted based on the experiments reported by Martin and Moyce (1952).

A rectangular column of water, in hydrostatic equilibrium, is confined between two vertical walls, as shown in figure 312.5. The water column is 1 unit wide and 2 unit high. At the beginning of the calculation, the right wall (dam) is removed and water is allowed to flow out along a dry horizontal floor.

The real-time position of the leading edge of the water is recorded during the experiment. This is a good test problem because it has simple boundary conditions and a simple initial configuration. 2D numerical models with two different types of mesh size ($\Delta x = 0.1m$ and $\Delta x = 0.05m$) are built (figure 312.6).

The comparison result between numerical solution, experiment result and benchmark solution by Hirt and Nichols (1981) is presented in figure 312.7.
Figure 312.6: Numerical model for validation of free surface flow

Figure 312.7: Validation result of free surface flow
It can be seen that numerical simulation matches well with both experiment result and benchmark solution. OpenFOAM, as a finite volume solver for free surface flow based on VOF method, is reliable.

312.3.2 Mass conservation verification

As mentioned in section 111.4.6, for a closed fluid system, mass conservation should be strictly satisfied. The total volume of fluid in the system can be calculated with equation 312.1.

\[ V_{total} = \sum_{i=1}^{n} \alpha_i V_i \]  \hspace{1cm} (312.1)

The time history record of total volume in the numerical example (section 312.2.1) is given in figure 312.8. It can be seen that the total volume remains almost constant during the simulation of SFI. In this example, after 400 time steps, the relative mass change is only 0.25%, which demonstrates that our coupling program has excellent performance regarding mass conservation.

![Figure 312.8: Time history record of total fluid volume](image)

312.3.3 SFI Verification & Validation

A box sloshing numerical experiment (rectangular tank under sway oscillations \( X(t) = Asin(wt) \)) is taken as the verification and validation test for solid fluid interaction. The box is 1.0 m long \((L)\) and 0.8 m height \((H)\). The depth \((D)\) of submerged water is 0.5 m. The natural frequency of the tank can
be calculated according to Lamb (1932):

$$\omega_i = \sqrt{\frac{g \pi i}{L} \tanh \left(\frac{\pi i D}{L}\right)} \quad i = 1, 2, 3, ...$$

(312.2)

where $\omega_i$ is the natural frequency, and $g$ is gravitational acceleration. The lowest linear mode $\omega_1$ is of primary importance for the tank sloshing phenomenon.

Based on the original analytical solution of Linton and McIver (2001), Jin et al. (2014) gives the equation for non-dimensional free surface elevation $\eta_{\text{max}}/A$ as shown in equation 312.3, where $b = L/2$, $\mu_m = (m + 0.5)\pi/b$, $K_m = \mu_m \tanh \lambda_m D$, $K = \omega^2/g$.

$$\frac{\eta_{\text{max}}}{A} = \frac{\omega}{gA} |A_0 b + \sum_{m=0}^{\infty} \left| \frac{2K}{\mu_m^2 b(K_m - K)} \right|$$

(312.3)

Jin et al. (2014) also conducted a 2D sloshing experiment and report detailed response of free surface elevation under different excitation magnitude $A$ and frequency $\omega$. Here $A$ is fixed as 2.5 mm and different frequency values ($\omega/\omega_1 = 0.5 \sim 2$) are adopted to implement verification and validation numerical test. The initial setup of numerical experiment can be seen in figure 312.9(a). Figure 312.9(b) and figure 312.9(c) are the sloshing response under excitation of first-mode ($\omega/\omega_1 = 1$) and third-mode ($\omega/\omega_3 = 1.793$) resonance frequency.

Figure 312.9: Numerical experiment of solid fluid interaction: (a) Initial setup (b) Sloshing response under first-mode resonance excitation (c) Sloshing response under third-mode resonance excitation

Compared with experiment record (figure 312.10) by Jin et al. (2014), it can be seen that the mode shape from our numerical simulation is same as the experimental observations.

The numerical results of $\eta_{\text{max}}/A$ are also plotted into the same figure as theoretical predication and experiment record, as shown in figure 312.11. Very good agreement can be observed, especially between numerical simulation and experiment result. This manifests that suitable mathematical equations about
Figure 312.10: Resonant wave shape: (a) First-order mode (b) Third-order mode reproduced from Jin et al. (2014)

SFI has been numerically solved in a correct way. The SFI coupling implementation in RealESSI is reliable.

Figure 312.11: Comparison among theoretical prediction, experiment observation and numerical simulation
Chapter 313

Quality Management System

313.1 Chapter Summary and Highlights

313.2 Reasoning Behind this Activity

Quality assurance for a numerical modeling system is of highest importance. A good, sound Quality Management System is therefore very important. This section provides details of quality management system used for the development and quality assurance (QA) for the Real-ESSI Simulator system (Jeremić et al., 1988-2021).
313.3 Real-ESSI Simulator System Quality Management System, based on ISO/IEC/IEEE 90003 Standard

This section is based on ISO/IEC/IEEE 90003 Developers et al. (2018). ISO/IEC/IEEE 9003 is a standard developed by the International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), and Institute of Electrical and Electronics Engineers (IEEE), as a guidance for the application of ISO 9001:2015 standard to the acquisition, supply, development, operation and maintenance of computer software and related support services.

313.3.1 Real-ESSI Simulator Developer Organization

313.3.1.1 Internal Issues

Internal issues that are relevant to the Real-ESSI Simulator quality management system:

- legal
- Technological
- Competitive
- Market
- Cultural
- Social
- Economic environment, local, national, international

313.3.1.2 Internal Issues

External issues that are relevant to the Real-ESSI Simulator quality management system:

- Values
- Culture
- Knowledge
- Performance
313.3.1.3 External and Internal Issues for the Real-ESSI Simulator

External and Internal issues for the Real-ESSI Simulator can include:

- Use of "cloud" services, that is beneficial for ease of access and business continuity, however needs research to ensure lowering all beneficial effects. The Real-ESSI Simulator is fully deployed on Amazon Web Services (AWS), and is tightly managed by development group as well as ESSI Consultants. AWS was chosen after in depth examining other cloud services, namely Google cloud, that actually provided no dedicated tightly parallel computers as of 2018, Microsoft Azure, that was fast but very expensive, and local parallel cloud, Real-ESSI Parallel Computer, that works well for developers, but requires much management for outside users.

- Use of personal computers, laptops, can create a problem in managing safety and security of Real-ESSI Simulator sources. Developers are aware of this issue and have agree do tight safety and security and have signed a licensing agreement that commits them to managing such safety and security.

- Risk of external attack on developers computers and network and on Real-ESSI Simulator deployed computers is controlled by regular, up to date Linux and AWS security.

- Delivery of the Real-ESSI Simulator is mostly handled through AWS, while there are a number of example of remote deployment, whereby remote users commit fully utilizing and following the Real-ESSI Simulator quality management system.

- Legal and operational issues for Real-ESSI Simulator use in contgext of safety, security and mission assurance.

313.3.1.4 Needs and expectations of Interested Parties

Interested parties that are relevant to Quality Management System and their requirements

- Customers, professional practice companies, require working system, that is efficient, easy to use and that can provide more optimal, more economic and more safe designs than what is currently available.

- Partners, collaborators, require in depth knowledge of the system, possibility to influence changes and additions to the Real-ESSI Simulator

- Staff, require stable and nourishing research and development environment
• DOE, not sure anymore?

• NRC, not sure anymore?

• UN-IAEA, ...

• Professional Practice: development of the system that improves design and assessment process, while maintaining practicality, achieving highly efficient, minimally disrupting training for use Real-ESSI

• Research Community: Contributions to modeling and simulations, and use of Real-ESSI to investigate new behavior of ESSI systems.

• Developers form the Real-ESSI Simulator group:

• Competitors:
  - SimCenter ?
  - French project?
  - Linear Elastic community, SASSI,
  - nonlinear FEM programs, LS-Dyna...

### 313.3.2 Scope of the Real-ESSI Simulator Quality Management System

Plan-do-Check-Act (PDCA) cycle

Risk-based thinking

**Quality Management Principles**
313.4 Real-ESSI Simulator System Quality Management System, based on ASME NQA-1 Standard

ASME Nuclear Quality Assurance (NQA-1) is a standard developed by the American Society of Mechanical Engineers (ASME) that provides quality assurance guidance and certification for organizations supplying items and services which provide a safety function for nuclear installations.

313.4.0.1 ASME NQA-1 for the Real-ESSI Simulator System
Chapter 314

Comparison with Other Programs

(2016-2017-2020-2021)

(In collaboration with Dr. Nima Tafazzoli, and Dr. Yuan Feng)
314.1 Chapter Summary and Highlights

314.1.1 Reasoning Behind this Activity

I personally do not like or approve code to code comparison. If results between two codes are the same, this does not prove that any tested code is right. If results between two codes are different, which code is right?

However, a number of professionals and industry in general feel very comfortable with some usually used codes/programs. In this sense, we provide code comparison with select widely used codes, to satisfy professionals. If results turn out to be the same (impossible to have exactly same results, but say very close) that is good for all codes involved, as we have a full verification suit and guaranty our accuracy to within limits of verification tests we used. If results are different, we still claim accuracy, as we have full verification suit and guaranty our accuracy to within limits of verification tests we used.
Part 400

Education, Training and Modeling,
Simulation Examples
Chapter 401

Ten Week Course on Nonlinear Finite Element Methods for Realistic Modeling and Simulation of Earthquakes, and Soils, and Structures, and their Interaction, Real-ESSI
401.1 Delivery

Instructor: Boris Jeremić, email: Jeremic@ucdavis.edu
Class meeting: two hour lecture/meetings, twice a week, flipped classroom method
Office hours: two hours, twice a week
Course delivery: live discussions and recorded lectures
Course WWW: http://sokocalo.engr.ucdavis.edu/~jeremic/Classes/ECI280A/

401.2 Objectives

This course will provide students with state of the art finite element methods, tools and models for solving elastic–plastic problems in geotechnical and structural engineering, with focus on Earthquakes, Soils, Structures and their Interaction (ESSI). Presented will be computational formulation, techniques and models for nonlinear, elastic-plastic finite element methods that are used in professional practice and research.

During this course students will:

- Learn about linear and nonlinear finite element modeling and simulation
- Select and calibrate nonlinear, elastic-plastic models for soil, rock, concrete, steel and interfaces
- Perform linear elastic and nonlinear, elastic-plastic analysis of solids and structures made of soil, rock, concrete, steel and their interfaces
- Perform elastic and nonlinear, inelastic analysis of soils-structure systems
- Become proficient in performing nonlinear analysis for soils, structures, interfaces, using different levels of sophistication, from simplified models to high fidelity elastic-plastic models of soil structure systems

Who Should Attend?
Students and practicing engineers who want to learn about and expand their knowledge of modeling and simulation for nonlinear/inelastic material behavior, for soils, rock, and structures,
401.3 Additional Information

Lecture Notes: http://sokocalo.engr.ucdavis.edu/~jeremic/LectureNotes/.

Computers: Most of the problems in this course will require numerical simulations. A finite element modeling system called Real-ESSI Simulator (http://real-essi.us) is available for computers running Windows, MacOS, Linux. Please refer to http://real-essi.us to find out how to use Real ESSI on local computers through Docker or on AWS computers. Other programs can be used as well, as long as they provide modeling and simulation capabilities that are required for assignments, example problems and term project.

Problems: Assigned weekly, students are expected to attempt to develop solutions. You are encouraged to discuss the approach to problem solutions with other students in the course as well as with the instructor.

Term Project: Term project will involve work related to developing or using numerical models for numerically simulating elastic–plastic problem of your choice, related to your interests. Term projects will be presented at the end of quarter.

Grading: term project 40%, final exam 60%.

Examination: Final exam: a week long, take home

Literature:


- Plasticity for Structural Engineers W. F. Chen and D. J. Han, Springer Verlag, 1988 ISBN 0-387-96711-7

- Boris Jeremić, Zhaohui Yang, Zhao Cheng, Guanzhou Jie, Nima Tafazzoli, Matthias Preisig, Panagiota Tasiopoulou, Federico Pisano, Jose Abell, Kohei Watanabe, Yuan Feng, Sumeet Kumar
401.4 Teaching Plan, Topics
401.4.1 Week I, Introduction

Introduction: Course objectives, methodology, computer modeling and simulation

Modeling and Simulation System Setup: Introduction to the Real-ESSI Simulator system. Computational Mechanics field of study, kinematics of deformation, strain, stress, linear and nonlinear elasticity, equilibrium relations, finite element method review, nonlinear analysis cycles;

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.1 on Page 1867 in Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture Notes: 101, 201, 205; Papers/Reports: CM988, CM2714, CM2715

Examples: Model development, finite element models, finite element mesh, boundary conditions, material models, loads, model verification process, linear and nonlinear elastic FEM with solids and structural elements,

Problems:

1.

2.

3.
401.4.2 Week II, Inelastic Finite Elements

**Theory:** Expanding the matrix deformation method, linear elastic truss element, Beams (Bernoulli, Timoshenko), solids, plates, walls (plane stress), shells, stiffness, forces, displacements, interpolating functions for displacements, internal and external forces, global equilibrium.

**Lectures:** Recorded lectures covering topics for this week can be found in Section 403.1.3 on Page 1869 in Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading:** Lecture Notes: 102; Papers/Reports:

**Examples:** Truss, beam, solid bricks, external forces, internal forces (sectional forces, stresses). Generalized nodal displacements and internal deformation (curvature, axial, shear, volumetric, general strains),

**Problems:**

1.

2.

3.
### 401.4.3 Week III, Micromechanics of Elasto-Plasticity

**Theory:** Micro-mechanical origins of elasto-plasticity, particles in contact, friction, Hertz, Mindlin-Deresiewicz contact/interface

**Lectures:** Recorded lectures covering topics for this week can be found in Section 403.1.4 on Page 1870 in Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading:** Lecture Notes: 103; Papers/Reports:

**Examples:** Particle contact problems.

**Problems:**

1. 
2. 
3. 
401.4.4 Week IV, Incremental Elastic-Plastic Theory

**Theory:** Incremental, continuum elasto-plasticity, Material Models, perfectly plastic, hardening and softening. Explicit, forward Euler and Implicit, backward Euler, constitutive integrations,

**Lectures:** Recorded lectures covering topics for this week can be found in Section 403.1.4 on Page 1870 in Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading:** Lecture Notes: 104; Papers/Reports:

**Examples:** Constitutive integrations, explicit and implicit computations: single element response using select elastic-plastic material models: von Mises, Drucker-Prager, Cam Clay. Perfectly plastic, isotropic hardening, kinematic hardening models and cyclic response. Inelastic, fiber (1D) and 3D structural models for concrete and steel.

**Problems:**

1. 
2. 
3. 
401.4.5 Week V, Inelastic, Elasto-Plastic Solids Modeling

**Theory:** Continuation: Incremental elasto-plasticity. Material modeling for practical applications. Advanced topics in constitutive elasto-plasticity, stability and accuracy, errors in constitutive integrations, problematic incremental steps, energy dissipation, sub-incrementation, line search, model calibrations.

**Lectures:** Recorded lectures covering topics for this week can be found in Section 403.1.4 on Page 1870 in Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading:** Lecture Notes: 104, 402, 403, 512; Papers/Reports:


**Problems:**

1. 
2. 
3. 

401.4.6 Week VI, Inelastic, Elastic-Plastic Interfaces, Joints, Contacts Modeling

**Theory:** Interface/Joint/Contact modeling: Hard contact, Soft contact. Axial contact stiffness, shear contact stiffness. Interface gap opening and closing. Saturated contacts, effective stress and buoyant forces on foundations.

**Lectures:** Recorded lectures covering topics for this week can be found in Section 403.1.4 on Page 1870 in Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading:** Lecture Notes: 104, 403, 512; Papers/Reports:

**Examples:** Interface: concrete to soil and rock, steel to soil and rock. Gap opening, closing. Shear interface, slip, no slip.

**Problems:**

1. 

2. 

3. 
401.4.7 Week VII, Inelastic, Elastic-Plastic Structures Modeling

**Theory:** Inelastic structural models, beams, plates, walls and shells.

**Lectures:** Recorded lectures covering topics for this week can be found in Section 403.1.4 on Page 1870 in Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading:** Lecture Notes: 102, 403, 512; Papers/Reports:

**Examples:** Nonlinear analysis of structures. Steel Frames. Reinforces concrete frames, walls, plates, shells.

**Problems:**

1.

2.

3.
401.4.8 Week VIII, Nonlinear Analysis Progress

Theory: Analysis Progress. Stages, increments, iterations, elastic–plastic stiffness matrix, pure incremental methods, force residuals, Newton iterative algorithm for finite element level iterations, constraints to the global (force residual) system of equations, equilibrium iterations, convergence, load control, displacement control, arc-length, hyper-spherical constraint, convergence criteria, automatic step size control, line search, stability and accuracy.

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.3 on Page 1869 in Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture Notes: 102, 107, 403; Papers/Reports:

Examples: Nonlinear analysis of structures and solids, elastic plastic solids, structures and contacts. Staged analysis steps, incremental only analysis with no equilibrium enforcement, incremental-iterative analysis, with equilibrium enforcement, convergence criteria (force, displacement), convergence tolerances, step size control.

Problems:

1.
2.
3.
401.4.9 Week IX, Verification and Validation

Theory: Verification, Validation and Prediction, basic theory, solution verification, manufactured solutions, validation experiments, prediction under uncertainty,

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.9 on Page 1877 in Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture Notes: 301, 302, 303..., 313, 314...; Papers/Reports:

Examples: Solution verification examples for elements, material models, constitutive integration algorithms, solution advancement algorithms.

Problems:

1. 
2. 
3.
401.4.10 Week X, Practical Considerations for Nonlinear Analysis

Theory: Elastic–plastic FEM modeling (practical recommendations for development and analysis of nonlinear (elastic-plastic) finite element models, phased development of general FEM (and ESSI in particular) models. Core Functionality for inelastic/nonlinear modeling, Energy dissipation

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.4 on Page 1870 and Section 403.1.5 on Page 1872 in Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture Notes: 510, 512; Papers/Reports:

Examples: Illustrations of algorithms and models described above, benefits and detriments of different algorithms and models.

Problems:

1.

2.

3.
Chapter 402

Ten Week Course on Dynamic Finite Element Methods for Realistic Modeling and Simulation of Earthquakes, and Soils, and Structures, and their Interaction, Real-ESSI

(1998-2021-)
402.1 Delivery

Instructor: Boris Jeremić, email: Jeremic@ucdavis.edu
Class meeting: two hour lecture/meetings, twice a week, flipped classroom method
Office hours: two hours, twice a week
Course delivery: live discussions and recorded lectures
Course WWW: http://sokocalo.engr.ucdavis.edu/~jeremic/Classes/ECI280B/

402.2 Objectives

This course will provide students with state of the art finite element methods, tools and models for solving dynamic problems in geotechnical and structural engineering, with focus on Earthquakes, Soils, Structures and their Interaction (ESSI). Presented will be computational formulation, techniques and models for linear and nonlinear, dynamic finite element methods that are used in professional practice and research.

During this course, students will:

- Learn about dynamic finite element modeling and simulation
- Develop dynamic modeling and simulations for linear and nonlinear soils and structures
- Perform dynamic, linear and nonlinear analysis of solids and structures made of soil, rock, concrete, steel and their interfaces, joints and contacts
- Develop and use of one component (1C), 3×1C and 3C seismic motions from given earthquake records and from analytic wave solutions
- Perform dynamic, nonlinear/inelastic earthquake soil structure interaction (ESSI) analysis
- Become proficient in performing nonlinear ESSI analysis using different levels of sophistication, from simplified models to high fidelity elastic-plastic ESSI models

Who Should Attend?

Students and practicing engineers who want to learn about and expand their knowledge of modeling and simulation for dynamic, nonlinear/inelastic material behavior, for soils, rock, and structures,
402.3 Additional Information

Lecture Notes: http://sokocalo.engr.ucdavis.edu/~jeremic/LectureNotes/.

Computers: Most of the problems in this course will require numerical simulations. A finite element modeling system called Real-ESSI Simulator (http://real-essi.us) is available for computers running Windows, MacOS, Linux. Please refer to http://real-essi.us to find out how to use Real ESSI on local computers through Docker or on AWS computers. Other programs can be used as well, as long as they provide modeling and simulation capabilities that are required for assignments, example problems and term project.

Problems: Assigned weekly, students are expected to attempt to develop solutions. You are encouraged to discuss the approach to problem solutions with other students in the course as well as with the instructor.

Term Project: Term project will involve work related to developing or using numerical models for numerically simulating elastic–plastic problem of your choice, related to your interests. Term projects will be presented at the end of quarter.

Grading: term project 40%, final exam 60%.

Examination: Final exam: a week long, take home

Literature:


- Plasticity for Structural Engineers W. F. Chen and D. J. Han, Springer Verlag, 1988 ISBN 0-387-96711-7

- Boris Jeremić, Zhaohui Yang, Zhao Cheng, Guanzhou Jie, Nima Tafazzoli, Matthias Preisig, Panagiota Tasiopoulou, Federico Pisano, Jose Abell, Kohei Watanabe, Yuan Feng, Sumeet Kumar
402.4 Teaching Plan, Topics
402.4.1 Week I, Introduction

Introduction: Course objectives, methodology, computer modeling and simulation; Dynamics of non-linear structures and soils during earthquakes, examples Preliminary theory, terminology, issues to be addressed: Deformation, kinematics of moving systems, elasticity, dynamic equilibrium relations, d’Alembert’s principle, forces in dynamic equilibrium, mass, damping, stiffness, external force, single degree of freedom systems,

Modeling and Simulation System Setup: Introduction to the Real-ESSI Simulator system. Computational Mechanics field of study, kinematics of deformation, strain, stress, linear and nonlinear elasticity, equilibrium relations, finite element method review, nonlinear analysis cycles;

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.1 on Page 1867 in Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture notes: 101, 102; Papers/Reports:

Examples: Model Development, simple models vs sophisticated models, pre-processing, post-processing, results visualization.

Problems:

1.
2.
3.
402.4.2 Week II, Dynamic FEM

**Dynamic FEM Theory:** Dynamic finite element method (FEM) equations, virtual work method in dynamics, nonlinear dynamic equations of motion, consistent and lumped mass, velocity and displacement proportional damping/energy dissipation, Rayleigh and Caughey viscous damping, linear and nonlinear material behavior.

**Lectures:** Recorded lectures covering topics for this week can be found in Section 403.1.3 on Page 1869 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading:** Lecture notes: 102; Papers/Reports:

**Examples:** Structural and solid elements and models, dynamic excitations, resonance, linear and nonlinear (elastic and inelastic/elastic-plastic) material models, viscous damping, consistent and lumped mass matrix.

**Problems:**

1.

2.

3.
402.4.3 Week III, Nonlinear FEM

Nonlinear FEM: Elasto-plasticity, material models for dynamics of soils and structures, material parameter calibration, uncertainty in material parameters, explicit and implicit constitutive integrations, Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.3 on Page 1869 and Section 403.1.4 on Page 1870 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
Reading: Lecture notes: 103, 104; Papers/Reports:
Examples: Elastic plastic solids, beams and shells, material energy dissipation, material damping
Problems:

1.
2.
3.
402.4.4 Week IV, Time Domain Nonlinear Dynamic FEM

**Time Domain Nonlinear Dynamic FEM**: Direct, time marching solution for dynamics of nonlinear, inelastic systems, general Newmark family of methods, stability and accuracy, nonlinear resonance, numerical damping, explicit and implicit algorithms, unconditionally and conditionally stable Newmark and Hilber–Hughes–Taylor \( \alpha \)-method, stability and accuracy, examples

**Lectures**: Recorded lectures covering topics for this week can be found in Section 403.1.3 on Page 1869 in Lecture Notes by Jeremić et al. (1989-2021) ([Lecture Notes URL](#)).

**Reading**: Lecture notes: 108; Papers/Reports:

**Examples**: Nonlinear solid and structural models direct time integration, step size, damping (material, viscous, numerical), stable and unstable computations.

**Problems**:

1. 
2. 
3. 
402.4.5 Week V, Earthquake Soil Structure Interaction (ESSI)

Earthquake Soil Structure Interaction (ESSI): Background, problem definition, seismic motions, seismic body and surface wave field, seismic energy propagation, free field motions, beneficial and detrimental effects, balancing input and dissipated energy.

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.8 on Page 1876 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture notes: 502; Papers/Reports:

Examples: Analytic development of ground motions, 3D vs 1D motions, seismic energy calculations.

Problems:

1.

2.

3.
402.4.6 Week VI, Seismic Motions

Seismic Motions: Free field vs ESSI motions, incoherent motions, Domain Reduction Method, boundary conditions, radiation damping, 3D inclined wave fields vs 1D vertical motions, nonlinear wave propagation simulations, time step size, element size, earthquake modeling.

Free field motions development, 1D motions, 3D/6D motions, regional scale models, Geophysical models,

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.7 on Page 1875 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture notes: 502, 511, 705, 706; Papers/Reports:

Examples: Real ESSI and analytic wave field models for free field and local (DRM) motions, element and time step size and propagation of (required) frequencies. Vertical and inclined waves development, and input into SSI models

Problems:

1. 

2. 

3. 
402.4.7 Week VII, Coupling with Internal and External Fluids

Dynamics of Coupling with Pore Fluids and External Fluids: Fully coupled, porous solid – pore fluid systems formulation, discretization, basic system of DOFs, coupling damping forces, specialization to slow (consolidation) and fast phenomena (ESSI, liquefaction), boundary conditions, initial conditions, stability and accuracy of various algorithms. Coupling with external fluids, pools, reservoirs.

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.3 on Page 1869 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture notes: 102, 505; Papers/Reports:

Examples: 1D and 3D coupled examples, consolidation, liquefaction and de-liquefaction waves, piles in liquefied soil... Coupling with external fluids, using OpenFOAM...

Problems:

1.

2.

3.
402.4.8 Week VIII, Dynamic Stochastic Elastic-Plastic FEM (SEPFEM)

Stochastic Elastic-Plastic Dynamic FEM:

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.6 on Page 1873 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture notes: Papers/Reports:

Examples:

Problems:

1.

2.

3.
402.4.9 Week IX, Verification and Validation

Verification and Validation: Definition, procedures, code verification, solution verification, validation experiments, model verification

Lectures: Recorded lectures covering topics for this week can be found in Section 403.1.9 on Page 1877 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

Reading: Lecture notes: 301-314; Papers/Reports:

Examples: modeling verification examples, verification for algorithms, elements. Availability of validation data.

Problems:

1.

2.

3.
402.4.10 Week X, ESSI Modeling and Simulation Synthesis

**ESSI Modeling and Simulation Synthesis**: Example building structure (boundary conditions, initial conditions, nonlinear contact (gap/slips), nonlinear soil/rock, 1D vs 3D seismic motions development, buoyant forces at foundation level, etc.)

**Lectures**: Recorded lectures covering topics for this week can be found in Section 403.1.11 on Page 1879 in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).

**Reading**: Lecture notes: 503, 504, 509 510, 512; Papers/Reports:

**Examples**: Real ESSI illustrative examples

**Problems**:

1.

2.

3.
Chapter 403
Online Education and Training

(In collaboration with Dr. Han Yang and Dr. Hexiang Wang)
403.1 Real-ESSI Simulator Online Education and Training

This chapter was created to present online material for the theory for modeling and simulation of earthquakes, soils, structures and their interaction, as well practical examples using the Real-ESSI Simulator, http://real-essi.us/.

It is worth nothing that some early recorded material for use of the Real-ESSI Simulator on Amazon Web Services was created in 2019, however, majority of presented, recorded material was created during Corona-Virus (COVID-19) pandemic and quarantine from March through June of 2020, in Zürich Switzerland, where Boris Jeremić was locked-up, and in Davis, California, where Han Yang and Hexiang Wang were locked-up... Internet worked very good across the Atlantic ocean, zoom.us worked really well as well. Development of online educational material continued with all three contributors now in Davis, California during Summer and Fall 2020, Winter and Spring 2021, still during partial/full lockdown, shelter in place, still using zoom.us, and still keeping physical distance, wearing face masks, etc.

In addition to organizing slides and vide lectures through this document, a youtube video channel for the Real-ESSI is available here: Real-ESSI youtube channel.

It is hoped that this material will be helpful to students and engineers that work in the area of modeling and simulation of earthquakes, soils, structures and their interaction.
403.1 Modeling and Simulations for ESSI

The following recorded lectures modeling and simulation approaches for Earthquakes, Soils, Structures and their Interaction are available:

403.1.1 Introduction to Modeling and Simulation

[PDF slides], and a direct [MP4 recording] and/or alternatively [YouTube video]

403.1.2 Introduction to Modeling Simplifications, Epistemic Uncertainty

[PDF slides], and a direct [MP4 recording] and/or alternatively [YouTube video]

403.1.3 Introduction to Parametric, Aleatory Uncertainty

[PDF slides], and a direct [MP4 recording] and/or alternatively [YouTube video]
403.1.2 Real-ESSI Simulator Modeling and Simulation System

The following recorded lectures about the Real-ESSI Simulator modeling and simulation system are available:

1. The Real-ESSI Simulator, Introduction:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

2. The Real-ESSI Simulator, Modeling Features:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

3. The Real-ESSI Simulator, Domain Specific Language:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

4. The Real-ESSI Simulator, Model Development:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

5. The Real-ESSI Simulator, Results Post Processing:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.3 Finite Element Method

The following recorded lectures on the finite element method are available:

403.1.3.1 Background

1. Introduction to the Finite Element Method (FEM):
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

2. Derivation of FEM equations of motions for single phase, dry material:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

3. Derivation of FEM equations of motions for coupled, two phase, fully and partially saturated material, u-p-U formulation:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.3.2 Nonlinear, Inelastic FEM

4. Nonlinear, Inelastic FEM, residual equations:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

5. Solution of nonlinear, inelastic FEM equations:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.3.3 Dynamic FEM

6. Dynamic FEM equations:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

7. Time marching algorithms for dynamic FEM equations:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.4 Deterministic Elasto-Plasticity

The following recorded lectures on deterministic elasto-plasticity are available:

403.1.4.1 Theory Background

1. Micromechanical origins of elasto-plasticity:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

2. Introduction to the incremental theory of elasto-plasticity:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

3. Explicit solution to the constitutive elastic-plastic problem:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

4. Implicit solution to the constitutive elastic-plastic problem:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.4.2 Elastic-Plastic Material Model Choices

5. Choice of elastic-plastic material models for soils and interfaces/contacts/joints:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

6. Choice of elastic-plastic material models for structural elements, beams and walls/plates/shells:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.4.3 Calibrating Elastic-Plastic Material Models

7. Calibration of elastic-plastic material models for sand:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

8. Calibration of elastic-plastic material models for clay:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

9. Calibration of elastic-plastic material models for interfaces/contacts/joints:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

10. Calibration of elastic-plastic material models for concrete, in reinforced beams and walls/plates/shells:
    PDF slides, and a direct MP4 recording and/or alternatively YouTube video
11. Calibration of elastic-plastic material models for steel, in reinforced beams and walls/beams/shells:

PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.5 Seismic Energy Dissipation

The following recorded lectures on energy dissipation are available:

403.1.5.1 Theory Background

1. Energy dissipation introduction:
   - [PDF slides](#), and a direct [MP4 recording](#) and/or alternatively [YouTube video](#)

2. Energy dissipation in solids:
   - [PDF slides](#), and a direct [MP4 recording](#) and/or alternatively [YouTube video](#)

3. Energy dissipation in fiber beams:
   - [PDF slides](#), and a direct [MP4 recording](#) and/or alternatively [YouTube video](#)

4. Energy dissipation in interfaces/joints/contacts:
   - [PDF slides](#), and a direct [MP4 recording](#) and/or alternatively [YouTube video](#)

5. Energy dissipation due to viscous effects:
   - [PDF slides](#), and a direct [MP4 recording](#) and/or alternatively [YouTube video](#)

6. Energy dissipation due to time integration, algorithmic, numerical effects:
   - [PDF slides](#), and a direct [MP4 recording](#) and/or alternatively [YouTube video](#)

403.1.5.2 Illustrative Examples

7. Energy dissipation examples:
   - [PDF slides](#), and a direct [MP4 recording](#) and/or alternatively [YouTube video](#)
403.1.6 Probabilistic Elasto-Plasticity and Stochastic Elastic-Plastic Finite Element Method

403.1.6.1 Theory Background

1. Introduction to the Polynomial Chaos (PC) expansion:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

2. Introduction to the Karhunen-Loève (KL) expansion:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

3. Introduction to the Stochastic Elastic-Plastic Finite Element Method (SEPFEM)
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

4. Introduction to Sensitivity Analysis
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.6.2 Choice and Calibration of Probabilistic Material Models and Probabilistic Seismic Loads

5. Choice, analysis and calibration of probabilistic elastic material parameters:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

6. Choice, analysis and calibration of probabilistic elastic-plastic, nonlinear, inelastic material parameters:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

7. Choice, analysis and calibration of probabilistic seismic motions:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.6.3 Simple Probabilistic Examples

8. SEPFEM, Two Examples:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

9. SEPFEM, Seismic Risk Analysis Example:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.6.4 Probabilistic Wave Propagation Examples

10. Analysis of one component (1C) seismic wave propagation with uncertain motions and uncertain elastic material parameters:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

11. Analysis of one component (1C) seismic wave propagation with uncertain motions and uncertain elastic-plastic, nonlinear, inelastic material parameters:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

12. Sensitivity analysis for uncertain motions and uncertain elastic material parameters:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

13. Sensitivity analysis for uncertain motions and uncertain elastic-plastic, nonlinear, inelastic material parameters:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.7 Seismic Motions

The following recorded lectures on seismic motions are available:

1. On earthquakes:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video

2. On six component (6C) seismic motions:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video

3. On the Domain Reduction Method (DRM):
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video

4. Development of DRM motions from surface records, 1C, $2 \times 1C$, and $3 \times 1C$:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video

5. Development of DRM motions from inclined, 3C seismic waves:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.8 Earthquake Soil Structure Interaction

The following recorded lectures on Earthquake Soil Structure Interaction (ESSI) are available:

1. On ESSI:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.9 Verification and Validation

Basic theory of Verification and Validation (V&V), as well as V&V examples for the Real ESSI Simulator are shown in recorded lectures below:

1. Verification and Validation introduction:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video

2. Real ESSI Simulator Verification and Validation system:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video

3. Real ESSI Simulator Verification examples:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video

4. Real ESSI Simulator Validation examples:
   - PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.10  High Performance Computing

High Performance Computing (HPC) is helping with analysis of sophisticated models efficiently on sequential and parallel computers.

403.1.10.1  HPC Introduction

1. HPC, an Introduction:
   
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

2. Fine grained HPC:
   
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

3. Coarse grained HPC, a Distributed Memory Parallel (DMP) Introduction:
   
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.10.2  HPC and Real-ESSI

1. Real-ESSI Simulator HPC Approach:
   
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

403.1.10.3  Real-ESSI Parallel Computing Examples

1. Real-ESSI Simulator Parallel Examples:
   
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video
403.1.11 Real-ESSI Simulator Examples

Select Real-ESSI examples are shown in recorded lectures below:

1. How to run already installed Real-ESSI program on a simple example:
   MP4 recording and/or alternatively YouTube video

2. Running Real-ESSI program for a frame model:
   MP4 recording and/or alternatively YouTube video

3. Running Real-ESSI program for a elastic-plastic solids model:
   MP4 recording and/or alternatively YouTube video

4. Running Real-ESSI program for a solids, beams and shells model:
   MP4 recording and/or alternatively YouTube video

5. Post-processing Real-ESSI results using Paraview for frame model:
   MP4 recording and/or alternatively YouTube video

6. Post-processing Real-ESSI results using Paraview for a solids, beams and shells model:
   MP4 recording and/or alternatively YouTube video

7. Developing a DRM SSI model, solids and beams:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video

8. Running a DRM SSI model, solids and beams:
   PDF slides, and a direct MP4 recording and/or alternatively YouTube video
Chapter 404

Constitutive, Material Behaviour Examples

(In collaboration with Dr. Yuan Feng and Dr. Han Yang)
404.1 Chapter Summary and Highlights

In this Chapter constitutive behavior of elastic-plastic material is illustrated through a number of examples.

All the examples described here, and many more, organized in sub-directories, for constitutive behavior, static and dynamic behavior can be directly downloaded from a repository at: http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_Examples/education_examples. These examples can then be tried, analyzed using Real-ESSI Simulator that is available on Amazon Web Services (AWS) computers around the word. Login to AWS market place and search for Real-ESSI...
404.2 Elastic Solid Constitutive Examples

404.2.1 Linear Elastic Constitutive Examples

404.2.1.1 Pure Shear, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type linear_elastic_isotropic_3d
   mass_density = 2E3 * kg/m^3
   elastic_modulus = 2E7 * Pa
   poisson_ratio= 0.25 ;
simulate constitutive testing strain control pure shear monotonic loading use ← material # 1
   confinement_strain = 0.001
   strain_increment_size = 0.0001
   number_of_increment = 100;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.1: Linear Elastic Pure Shear Monotonic Loading
404.2.1.2 Pure Shear, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type linear_elastic_isotropic_3d
    mass_density = 2E3 * kg/m^3
    elastic_modulus = 2E7 * Pa
    poisson_ratio= 0.25 ;
simulate constitutive testing strain control pure shear cyclic loading use ←
    material # 1
    confinement_strain = 0.001
    strain_increment_size = 0.0001
    maximum_strain = 0.01
    number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.2: Linear Elastic Pure Shear Cyclic Loading.
404.2.1.3 Uniaxial Strain, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type linear_elastic_isotropic_3d
  mass_density = 2E3 * kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio= 0.0;
simulate constitutive testing strain control uniaxial monotonic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  number_of_increment = 100;
baby;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.3: Linear Elastic Uniaxial Monotonic Loading
404.2.1.4 Uniaxial Strain, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```
model name "test";
add material # 1 type linear_elastic_isotropic_3d
  mass_density = 2E3 * kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio= 0.25 ;
simulate constitutive testing strain control pure shear cyclic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  maximum_strain = 0.01
  number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.4: Linear Elastic Uniaxial Cyclic Loading
404.2.2 Nonlinear Elastic Constitutive Examples

404.2.2.1 Triaxial Uniform Pressure, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The Duncan-Chang nonlinear elastic materials:

\[ E = K p_a \left( \frac{\sigma_3}{p_a} \right)^n \]  

(404.1)

where \( K \) and \( n \) are material constants. And pressure \( p_a \) is atmospheric pressure. And stress \( \sigma_3 \) is the minor principal stress.

Material properties in Real-ESSI input:

```plaintext
1 model name "test";
2 add material # 1 type Duncan_Chang_nonlinear_elastic_isotropic_3d_LT
3   mass_density = 2E3 * kg/m^3
4   initial_elastic_modulus = 3E5 * Pa
5   poisson_ratio= 0.15
6   DuncanChang_K = 1E3
7   DuncanChang_pa = 1E5 * Pa
8   DuncanChang_n = 0.5 ;
9 simulate constitutive testing strain control triaxial confinement loading use ←
10   material # 1
11   strain_increment_size = 0.00001
12   maximum_strain = 0.01
13   number_of_increment = 2000;
14 bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.5: Results of Duncan-Chang Nonlinear Elastic Monotonic Loading
404.2.3 Nonlinear Elastic Constitutive Examples

404.2.3.1 Triaxial Uniform Pressure, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type Duncan_Chang_nonlinear_elastic_isotropic_3d_LT
    mass_density = 2E3 * kg/m^3
    initial_elastic_modulus = 3E5 * Pa
    poisson_ratio= 0.15
    DuncanChang_K = 1E3
    DuncanChang_pa = 1E5 * Pa
    DuncanChang_n = 0.5 ;
simulate constitutive testing strain control triaxial confinement loading use ←
    material # 1
    strain_increment_size = 0.00001
    maximum_strain = 0.01
    number_of_increment = 2000;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.6: Results of Duncan-Chang Nonlinear Elastic Cyclic Loading
404.3 Elastic Plastic Solid Constitutive Examples

404.3.1 Elastic Perfectly Plastic Constitutive Examples

404.3.1.1 Pure Shear

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio=0.25
  von_mises_radius = 1E5*Pa
  kinematic_hardening_rate = 0.0*Pa
  isotropic_hardening_rate = 0.0*Pa;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear cyclic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  maximum_strain = 0.01
  number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.7: Perfectly Plastic Pure Shear Cyclic Loading.
404.3.1.2 Uniaxial Strain

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
mass_density = 2E3*kg/m^3
elastic_modulus = 2E7 * Pa
poisson_ratio=0.25
von_mises_radius = 1E5*Pa
kinematic_hardening_rate = 0.0 *Pa
isotropic_hardening_rate = 0.0*Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
yield_function_relative_tolerance = 1E-2
stress_relative_tolerance = 1E-3
maximum_iterations = 30;
simulate constitutive testing strain control uniaxial cyclic loading use ←
material # 1
confinement_strain = 0.001
strain_increment_size = 0.0001
maximum_strain = 0.01
number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.8: Perfectly Plastic Uniaxial Cyclic Loading
404.3.2 Elastic Plastic, Isotropic Hardening, Constitutive Examples

404.3.2.1 Pure Shear, Monotonic Loading

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio=0.25
  von_mises_radius = 1E5*Pa
  kinematic_hardening_rate = 0.0*Pa
  isotropic_hardening_rate = 2E6 *Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear monotonic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  number_of_increment = 99;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](#)

Figure 404.9: Isotropic Hardening Pure Shear Monotonic Loading
### 404.3.2.2 Pure Shear, Cyclic Loading

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio=0.25
  von_mises_radius = 1E5*Pa
  kinematic_hardening_rate = 0.0*Pa
  isotropic_hardening_rate = 2E6 *Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear cyclic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  maximum_strain = 0.01
  number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](#)

Figure 404.10: Isotropic Hardening Pure Shear Cyclic Loading.
404.3.2.3 Uniaxial Strain, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio = 0.25
  von_mises_radius = 5E4*Pa
  kinematic_hardening_rate = 0.0*Pa
  isotropic_hardening_rate = 2E6 *Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control uniaxial monotonic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  number_of_increment = 99;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.11: Isotropic Hardening Uniaxial Monotonic Loading
404.3.2.4 Uniaxial Strain, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
    mass_density = 2E3*kg/m^3
    elastic_modulus = 2E7 * Pa
    poisson_ratio=0.25
    von_mises_radius = 5E4*Pa
    kinematic_hardening_rate = 0.0*Pa
    isotropic_hardening_rate = 2E6 *Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
    yield_function_relative_tolerance = 1E-2
    stress_relative_tolerance = 1E-3
    maximum_iterations = 30;
simulate constitutive testing strain control uniaxial cyclic loading use ←
    material # 1
    confinement_strain = 0.001
    strain_increment_size = 0.0001
    maximum_strain = 0.01
    number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.12: Isotropic Hardening Uniaxial Cyclic Loading
404.3.3 Elastic Plastic, Kinematic Hardening, Constitutive Examples

404.3.3.1 Pure Shear, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
elastic_modulus = 2E7 * Pa
poisson_ratio=0.25
von_mises_radius = 1E5*Pa
kinematic_hardening_rate = 2E6*Pa
isotropic_hardening_rate = 0.0*Pa;
define NDMaterial constitutive integration algorithm Backward.Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear monotonic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  number_of_increment = 99;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image.png)

Figure 404.13: Kinematic Hardening Monotonic Cyclic Loading
404.3.3.2 Pure Shear, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio=0.25
  von_mises_radius = 1E5*Pa
  kinematic_hardening_rate = 2E6*Pa
  isotropic_hardening_rate = 0.0*Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear cyclic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  maximum_strain = 0.01
  number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.14: Kinematic Hardening Pure Shear Cyclic Loading.
404.3.3.3 Uniaxial Strain, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio=0.25
  von_mises_radius = 5E4*Pa
  kinematic_hardening_rate = 2E6*Pa
  isotropic_hardening_rate = 0.0*Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control uniaxial monotonic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  number_of_increment = 99;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.15: Kinematic Hardening Uniaxial Monotonic Loading
404.3.3.4 Uniaxial Strain, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "test";
add material # 1 type VonMises
  mass_density = 2E3*kg/m^3
  elastic_modulus = 2E7 * Pa
  poisson_ratio=0.25
  von_mises_radius = 5E4*Pa
  kinematic_hardening_rate = 2E6*Pa
  isotropic_hardening_rate = 0.0*Pa ;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-2
  stress_relative_tolerance = 1E-3
  maximum_iterations = 30;
simulate constitutive testing strain control uniaxial cyclic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  maximum_strain = 0.01
  number_of_cycles = 1;
bye;
```

Material Response:

![Material Behavior: Stress-Strain](image)

Figure 404.16: Kinematic Hardening Uniaxial Cyclic Loading
404.3.4 Elastic Plastic, Armstrong-Frederick, von-Mises, Constitutive Examples

404.3.4.1 Pure Shear, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```plaintext
model name "vmaf";
add material # 1 type vonMisesArmstrongFrederick
  mass_density = 0.0*kg/m^3
  elastic_modulus = 2E7*N/m^2
  poisson_ratio = 0.0
  von_mises_radius = 100 * Pa
  armstrong_frederick_ha = 2E7*N/m^2
  armstrong_frederick_cr = 1000
  isotropic_hardening_rate = 0*Pa;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-6
  stress_relative_tolerance = 1E-6
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear cyclic loading use <-
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  maximum_strain = 0.01
  number_of_cycles = 1;
bye;
```

Material Response:

![Stress-Strain](image)

Figure 404.17: Material von-Mises Armstrong-Frederick under Pure Shear Cyclic Loading.
404.3.5 Elastic Plastic, Armstrong-Frederick, Drucker-Prager, Constitutive Examples

404.3.5.1 Pure Shear, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Material properties in Real-ESSI input:

```
model name "test";
phi = 5;
phirad = pi*phi/180;
eta = 6*sin(phirad)/(3-sin(phirad));
add material # 1 type DruckerPragerNonAssociateArmstrongFrederick
  mass_density = 0.0*kg/m^3
  elastic_modulus = 2E7*N/m^2
  poisson_ratio = 0.0
  druckerprager_k = eta
  armstrong_frederick_ha = 2E7*N/m^2
  armstrong_frederick_cr = 100
  isotropic_hardening_rate = 0*Pa
  initial_confining_stress = 1*Pa
  plastic_flow_xi = 0.0
  plastic_flow_kd = 0.0;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-6
  stress_relative_tolerance = 1E-6
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear cyclic loading use ←
  material # 1
  confinement_strain = 0.001
  strain_increment_size = 0.0001
  maximum_strain = 0.01
  number_of_cycles = 1;
bye;
```

Material Response:
Figure 404.18: Drucker-Prager Armstrong-Frederick under Pure Shear Cyclic Loading.

404.3.6 Elastic Plastic, SanSAND, Constitutive Examples

404.3.6.1 Bardet Constraint Examples

The compressed package of Real-ESSI input files and postprocessing scripts and results for this example is available HERE. Material Response is shown in Figure 404.19.

Figure 404.19: SaniSAND response.
404.4 Stiffness Reduction and Damping Curves Modeling

404.4.1 Multi-yield-surface von-Mises

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

404.4.1.1 Model description

This model illustrates the $G/G_{\text{max}}$ input to multi-yield-surface von-Mises material. This example is based on one Gauss-point with multi-yield-surface von-Mises material. The $G/G_{\text{max}}$ is converted to material modeling parameters (yield-surface size and hardening parameter) inside the DSL.

404.4.1.2 Real-ESSI input file

```plaintext
model name "test";
add material #1 type vonMisesMultipleYieldSurfaceGoverGmax
  mass_density = 0.0*kg/m^3
  initial_shear_modulus = 3E8 * Pa
  poisson_ratio = 0.0
  total_number_of_shear_modulus = 9
  GoverGmax = "1,0.995,0.966,0.873,0.787,0.467,0.320,0.109,0.063"
  ShearStrainGamma = "0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01"
);
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-6
  stress_relative_tolerance = 1E-6
  maximum_iterations = 30
  incr_size = 0.000001;
  max_strain = 0.005;
  num_of_increm = max_strain/incr_size -1;
simulate constitutive testing strain control pure shear use material # 1
  confinement_strain = 0.0
  strain_increment_size = incr_size
  maximum_strain = max_strain
  number_of_increment = num_of_increm;
bye;
```

Material Response at Gauss Point:

Computed $G/G_{\text{max}}$ curve exactly matches the one used for input at control points.

The difference in $G/G_{\text{max}}$ between control points can be reduced by using more than just 9 control points as in this example.
Material Behavior: Stress-Strain

**Figure 404.20:** Stress-Strain Relationship

Multi-Yield-Surface vonMises G/Gmax

**Figure 404.21:** The G/Gmax results.
Figure 404.22: Damping Ratio Plot
404.4.2 Multi-yield-surface Drucker-Prager

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

404.4.2.1 Problem description

This model illustrates the G/Gmax input to multi-yield-surface Drucker-Prager material. Purely deviatoric plastic flow is used in this material, which means that the parameter dilation_scale is set to zero. If user wants to model change of volume (dilation or compression) for this material, then G/Gmax curve need to be iterated upon manually by changing yield surface size directly, which is done using different DruckerPragerMultipleYieldSurface command. This example is based on one Gauss-point which use multi-yield-surface Drucker-Prager material. The G/Gmax is converted to the yield-surface size and hardening parameter inside the DSL.

404.4.2.2 Real-ESSI input file:

```plaintext
model name "test";

add material # 1 type DruckerPragerMultipleYieldSurfaceGoverGmax
  mass_density = 0.0*kg/m^3
  initial_shear_modulus = 3E8 * Pa
  poisson_ratio = 0.0
  initial_confining_stress = 1E5 * Pa
  reference_pressure = 1E5 * Pa
  pressure_exponential_n = 0.5
  cohesion = 0. * Pa
  dilation_angle_eta =1.0
  dilation_scale = 0.0
  total_number_of_shear_modulus = 9
  GoverGmax =
    "1,0.995,0.966,0.873,0.787,0.467,0.320,0.109,0.063"
  ShearStrainGamma =
    "0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01"
;
define NDMaterial constitutive integration algorithm Backward_Euler
  yield_function_relative_tolerance = 1E-6
  stress_relative_tolerance = 1E-6
  maximum_iterations = 30;
simulate constitutive testing strain control pure shear use material # 1
  confinement_strain = 0.0
  strain_increment_size = 0.000001
  maximum_strain = 0.005
  number_of_increment = 0.005 / 0.000001 -1 ;
bye;
```
Inside the DSL, the yield surface radius is calculated as $\sqrt{3}\sigma_y$, where $\sigma_y$ is the yield stress of the corresponding yield surface. Then, the radius is divided by the confinement to obtain the slope (opening angle).

The hardening parameter is calculated as

$$\frac{1}{H'_i} = \frac{1}{H_i} - \frac{1}{2G}$$

(404.2)

where $H'_i$ is the current hardening parameter corresponding to yield surface $i$. $H_i$ is the current tangent shear modulus to surface $i$, namely, $H_i = 2(\tau_{i+1} - \tau_i)/(\gamma_{i+1} - \gamma_i)$. And $G$ is the initial shear modulus.
Figure 404.24: Nested-Yield-Surface Drucker-Prager G/Gmax results

Figure 404.25: Damping Ratio Plot
404.4.3 Simulate Stiffness Reduction using von-Mises Armstrong-Frederick

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

404.4.3.1 Model description

This model illustrates the simulation of stiffness reduction using von-Mises Armstrong-Frederick. This example is based on one Gauss-point.

404.4.3.2 Real-ESSI input file:

```plaintext
model name "test";
add material # 1 type vonMisesArmstrongFrederick
    mass_density = 2500.0*kg/m^3
    elastic_modulus = 3E7*N/m^2
    poisson_ratio = 0.2
    von_mises_radius = 300 * Pa
    armstrong_frederick_ha = 5*3E7*N/m^2
    armstrong_frederick_cr = 25000
    isotropic_hardening_rate = 0*Pa
;
define NDMaterial constitutive integration algorithm Backward_Euler
    yield_function_relative_tolerance = 1E-6
    stress_relative_tolerance = 1E-6
    maximum_iterations = 30
;
incr_size = 0.000001;
max_strain= 0.005;
num_of_increm = max_strain/incr_size -1;
simulate constitutive testing strain control pure shear use material # 1
    confinement_strain = 0.0
    strain_increment_size = incr_size
    maximum_strain = max_strain
    number_of_increment = num_of_increm;
bye;
```

The von-Mises Armstrong-Frederick material behavior matches the stiffness reduction curve.
Figure 404.26: The stiffness reduction results.

Figure 404.27: Damping Ratio Plot
404.5 Cosserat, Micropolar Material Modeling
404.5.1 Cosserat, Micropolar Elastic Material Model (example in development)
404.5.2 Cosserat, Micropolar Elastic-Plastic von Mises Material Model (example in development)
404.5.3 Cosserat, Micropolar Elastic-Plastic Drueckr Drueckr Prager Material Model (example in development)
Chapter 405

Static Examples

(In collaboration with Prof. José Abell, Dr. Yuan Feng, Mr. Sumeet Kumar Sinha, and Dr. Han Yang)
405.1 Chapter Summary and Highlights

In this Chapter static modeling and simulation of solids and structures is illustrated through a number of examples.

All the examples described here, and many more, organized in sub-directories, for constitutive behavior, static and dynamic behavior can be directly downloaded from a repository at: http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_Examples/education_examples. These examples can then be tried, analyzed using Real-ESSI Simulator that is available on Amazon Web Services (AWS) computers around the word. Login to AWS market place and search for Real-ESSI...

405.2 Static Elastic Solid Examples

405.2.1 Statics, Bricks, with Nodal Forces

405.2.1.1 Statics, 8 Node Brick, with Nodal Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with a nodal force at the tip. Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.0$. The force direction was shown in Figure (405.1).

![Figure 405.1: Problem description for cantilever beams.](image)

The mesh is generated with elastic 8 node brick.
405.2.1.2 Statics, 27 Node Brick, with Nodal Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with a nodal force at the tip. Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \). The force direction was shown in Figure (405.3).

The mesh is generated with elastic 27 node brick.

405.2.1.3 Statics, 8-27 Node Brick, with Nodal Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.
Problem description: a cantilever with a nodal force at the tip. Length=2m, Width=2m, Height=2m, $\nu = 0.0$. The force direction was shown in Figure (405.5).

![Figure 405.5: Problem description for cantilever beams.](image)

The mesh is generated with an elastic 8-27 node brick. As shown in the Figure 405.16, some of the nodes are missing on purpose. The variable node brick element is usually used as the transition mesh between 8 node brick and 27 node brick.

![Figure 405.6: One 8-27 Node elements.](image)

405.2.2 Statics, Bricks, with Surface Loads

405.2.2.1 Statics, 8 Node Brick, with Surface Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with the load on one surface. Length=2m, Width=2m, Height=2m, $\nu = 0.0$. The force distribution was shown in Figure (405.7).

The mesh is generated with an elastic 8 node brick.
405.2.2.2 Statics, 27 Node Brick, with Surface Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with the load on one surface. Length=2m, Width=2m, Height=2m.

The force distribution was shown in Figure (405.9).

The mesh is generated with an elastic 27 node brick.
405.2.3 Statics, Bricks, with Body Forces

405.2.3.1 Statics, 8 Node Brick, with Body Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with self weight on the whole element. Length=6m, Width=1m, Height=1m, $\nu = 0.3$. The force direction was shown in Figure (405.11).

The mesh is generated with an elastic 8 node brick.

---

Figure 405.10: One element with surface load.

Figure 405.11: Problem description for cantilever beams.

Figure 405.12: Six 8NodeBrick elements.
405.2.3.2 Statics, 27 Node Brick, with Body Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with self weight on the whole element. Length=6m, Width=1m, Height=1m, \( \nu = 0.3 \). The force direction was shown in Figure (405.13).

![Figure 405.13: Problem description for cantilever beams.](image)

The mesh is generated with an elastic 27 node brick.

![Figure 405.14: Six 27NodeBrick elements.](image)
405.2.3.3 Statics, 8-27 Node Brick, with Body Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with self weight on the whole element. Length=2m, Width=2m, Height=2m, $\nu = 0.3$. The force direction was shown in Figure (405.15).

![Figure 405.15](image1.png)

Figure 405.15: Problem description for cantilever beams.

The mesh is generated with an elastic 8-27 node brick. As shown in the Figure 405.16, some of the nodes are missing on purpose. The variable node brick element is usually used as the transition mesh between 8 node brick and 27 node brick.

![Figure 405.16](image2.png)

Figure 405.16: One variable Node Brick elements.
405.3 Static Elastic Structural Examples

405.3.1 Statics, Truss, with Nodal Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with the nodal load on the tip. Length=1m, Cross Section=1m². The cross section shape is not necessarily a square. The force direction was shown in Figure (405.17). Truss only takes axial force.

![Figure 405.17: Problem description for a cantilever.](image-url)
405.3.2 Statics, Elastic Beam, with Nodal Forces

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Problem description: a cantilever with nodal load on the tip. Length=1m, Width=1m, Height=1m, $E=1\, Pa$. The force direction was shown in Figure (405.18).

![Figure 405.18: Problem description for cantilever beams.](#)
405.3.3 Statics, Elastic Beam, with Body Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with self weight. Length=1m, Width=1m, and Height=1m. The force direction was shown in Figure (405.19).

![Figure 405.19: Problem description for cantilever beams.](image)
405.3.4 Statics, ShearBeam Element

405.3.4.1 Problem description

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

In the element type "ShearBeam", only one Gauss point exists. ShearBeam element was used here to test the von Mises Armstrong-Frederick material model. Vertical force $F_z$ was used to apply confinement to the element. Then, cyclic force $F_x$ is used to load. Usually, pressure-dependent materials, like Drucker-Prager, require the confinement. The pressure-independent materials, like von Mises, do not require the confinement.

![ShearBeam element](image)

Figure 405.20: ShearBeam element.

405.3.4.2 Results

Resulting stress-strain relationship is shown in Fig.(707.51).
Material Behavior: Stress-Strain

Figure 405.21: Shear stress-strain response.
405.3.5 Statics, Elastic Shell, with Nodal Forces

405.3.5.1 ANDES Shell, out of Plane Force

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, $E=1E8\text{Pa}$, $\nu=0.0$.

The force direction was shown in Figure (405.22).

![Figure 405.22: Problem description for cantilever beams.](image)

405.3.5.2 Perpendicular to Plane (bending)

The mesh and the out-of-plane force is shown in Fig. 405.23.

![Figure 405.23: Six 4NodeANDES elements.](image)

405.3.5.3 ANDES Shell, In-plane Force

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with a nodal force at the tip. Length=6m, Width=1m, Height=1m, Force=100N, $E=1E8\text{Pa}$, $\nu=0.0$. The force direction was shown in Figure (405.24).

The mesh and the inplane force is shown in Fig. 405.25.
Figure 405.24: Problem description for cantilever beams.

Figure 405.25: Six 4NodeANDES elements.
405.3.6 Statics, Elastic Shell, with Body Forces

405.3.6.1 ANDES shell under the out-of-Plane Body Force

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: Length=6m, Width=1m, Height=1m, Force=100N, $E=1E8Pa$, $\nu = 0.0$. The force direction was shown in Figure (405.26).

![Figure 405.26: Problem description for cantilever beams.](image)

405.3.6.2 ANDES Shell, In-plane Body Force

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Problem description: a cantilever with a nodal force at the tip. Length=6m, Width=1m, Height=1m, Force=100N, $E=1E8Pa$, $\nu = 0.0$. The force direction was shown in Figure (405.28).

![Figure 405.27: Six 4NodeANDES elements.](image)
Figure 405.28: Problem description for cantilever beams.

Figure 405.29: Six 4NodeANDES elements.
405.4 Statics, Interface/Contact Elements

405.4.1 Statics, Two Bar Normal Interface/Contact Problem Under Monotonic Loading.

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

This is an example of normal monotonic loading on a 1-D contact/interface between two bars separated by an initial gap of 0.1 unit. An illustrative diagram of the problem statement is shown below.

![Diagram of Two Bar Normal Interface/Contact Problem under monotonic loading with initial gap.](image)

Figure 405.30: Illustration of Two Bar Normal Interface/Contact Problem under monotonic loading with initial gap.

The displacement output of Node 2 and Node 3 are shown below.

![Graph of displacement vs. force for Nodes 2 and 3.](image)

Figure 405.31: Displacement of Nodes 2 and 3.
405.4.2 Statics, Four Bar Interface/Contact Problem With Normal and Shear Force Under Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

This is an example to show the normal and tangential behavior (stick and slip case) of contacts using four bars in 2-D plane. The bars in x-directions are in contact/interface (initial gap=0).

![Figure 405.32: Illustration of Four Bar Normal Interface/Contact Problem With Normal and Shear Force Under Monotonic Loading with no initial gap.](image_url)

Figure 405.32: Illustration of Four Bar Normal Interface/Contact Problem With Normal and Shear Force Under Monotonic Loading with no initial gap.
The displacement output of Node 2 and Node 3 are shown below.

Figure 405.33: Displacement of Nodes 2 and 3 along y direction.
405.4.3 Statics, 3-D Truss example with normal confinement and Shear Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

A simple 3-D truss example with Normal confinement in z-direction of $F_N = 0.5N$, friction coefficient $\mu = 0.2$ and shear loading of magnitude $F_s = 0.5N$. Figure 707.57 below, shows the description of the problem.

![Diagram of 3-D Truss Problem with confinement loading in z-direction of 0.5N and then shear loading of 0.5N in x-y plane.](image)

Figure 405.34: Illustration of 3-D Truss Problem with confinement loading in z-direction of 0.5N and then shear loading of 0.5N in x-y plane.

The generalized displacement response of the tangential loading stage is shown below.

![Graphs showing displacement in x and y directions](image)

Figure 405.35: Displacements of Node 2 with applied shear tangential load step.
Figure 405.36: Resisting force by the contact/interface element with applied shear tangential load step.
405.4.4 Statics, Six Solid Blocks Example With Interface/Contact

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

This is a 3-D solid block example with initial normal and then tangential load on different surfaces as shown below.

![Figure 405.37: Illustration of Six Solid Blocks Example with Interface/Contact with first normal and then tangential loading stages.](image)

The generalized displacement field of the two loading stages normal loading and tangential loading is shown below.

![Figure 405.38: Generalized displacement magnitude visualization of normal loading.](image)
405.5 Static Inelastic Solid Examples

405.5.1 Statics, Bricks, Elastic-Plastic, von Mises, with Nodal Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Figure 405.40: Perfectly Plastic Pure Shear Cyclic Loading.

Material Response at Gauss Point:

405.5.2 Statics, Bricks, Elastic-Plastic, Drucker Prager, with Nodal Forces

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.
Material Behavior: Stress-Strain

Figure 405.41: Results of Perfectly Plastic Pure Shear Cyclic Loading.

Model Description:

Figure 405.42: Diagram of Drucker-Prager Armstrong-Frederick Pure Shear Cyclic Loading.

Material Response at Gauss Point:
Figure 405.43: Result of Drucker-Prager Armstrong-Frederick Pure Shear Cyclic Loading.
405.6 Static Inelastic Shell Examples (example in development)

405.7 Statics, Elastic Single Solid Finite Finite Element Examples

405.7.1 Statics, Linear Elastic, Solid Examples

405.7.1.1 Statics, Pure Shear, Monotonic Loading

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Model Description:

![Diagram Linear Elastic Solid Pure Shear Monotonic Loading.](image)

Figure 405.44: Diagram Linear Elastic Solid Pure Shear Monotonic Loading.

Material Response at Gauss Point:

![Material Behavior: Stress-Strain](image)

Figure 405.45: Results of Linear Elastic Solid Pure Shear Monotonic Loading.
405.7.1.2 Pure Shear, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

![Diagram Linear Elastic Solid Pure Shear Cyclic Loading.](image)

Figure 405.46: Diagram Linear Elastic Solid Pure Shear Cyclic Loading.

Material Response at Gauss Point:

![Material Behavior: Stress-Strain](image)

Figure 405.47: Results of Linear Elastic Solid Pure Shear Cyclic Loading.

405.7.1.3 Uniaxial Strain, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Material Response at Gauss Point:
405.7.1.4 Uniaxial Strain, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:
Material Response at Gauss Point:

405.7.2 Statics, Nonlinear Elastic, Duncan-Chang, Pure Shear, Solid Examples

405.7.2.1 Pure Shear, Monotonic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.
Figure 405.50: Linear Elastic Uniaxial Strain Cyclic Loading.

Figure 405.51: Results of Linear Elastic Pure Shear Cyclic Loading.

Model Description:
Material Response at Gauss Point:

405.7.2.2 Pure Shear, Cyclic Loading

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.
Figure 405.52: Nonlinear Elastic Uniaxial Strain Monotonic Loading.

Figure 405.53: Results of Nonlinear Elastic Pure Shear Monotonic Loading.

Model Description:

Material Response at Gauss Point:
405.8 Statics, Elastic-Plastic Single Solid Finite Element Examples

405.8.1 Statics, Elastic Perfectly Plastic, Cyclic Loading, Pure Shear Solid Examples

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Material Response at Gauss Point:
405.8.1.1 Statics, von Mises Yield Function, Isotropic Hardening

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Material Response at Gauss Point:
405.8.1.2 Statics, von Mises Yield Function, Kinematic Hardening

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Material Response at Gauss Point:
405.8.1.3 Statics, Drucker Prager Yield Function, von-Mises Plastic Potential Function, Perfectly Plastic Hardening Rule

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:
405.8.1.4 Statics, Drucker Prager Yield Function, Drucker Prager Plastic Potential Function, Perfectly Plastic Hardening Rule

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:
Material Response at Gauss Point:
Figure 405.64: Pure Shear Cyclic Loading.

Figure 405.65: Results of Associative Drucker Prager Pure Shear Cyclic Loading.

405.8.2 Statics, Drucker Prager with Armstrong Frederick Nonlinear Kinematic Hardening Material Model

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Material Response at Gauss Point:
Figure 405.66: Pure Shear Cyclic Loading.

Figure 405.67: Result of Drucker-Prager Armstrong-Frederick Pure Shear Cyclic Loading.
405.9 Statics, Elastic, Fiber Cross Section Beam Finite Element Examples

405.9.1 Statics, Linear Elastic, Normal Loading and Pure Bending Fiber Cross Section Beam Finite Element Examples

405.9.1.1 Linear Elastic Normal Loading, (Fiber Cross Section) Beam Finite Element Examples

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The linear elastic beam is represented by the elastic section. This example is under the load of normal loading.

![Normal Loading on the Fiber Beam with Elastic Section](image)

**Figure 405.68**: Normal Loading on the Fiber Beam with Elastic Section.

The elastic section represents the cross section properties of the beam.

![Diagram of the Fiber Beam with Elastic Section](image)

**Figure 405.69**: Diagram of the Fiber Beam with Elastic Section.
405.9.1.2 Linear Elastic Pure Bending, (Fiber Cross Section) Beam Finite Element Examples

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The linear elastic beam is represented by the elastic section. This example is under the bending load.

![Bending on the Fiber Beam with Elastic Section](image1)

Figure 405.70: Bending on the Fiber Beam with Elastic Section.

The elastic section represents the cross section properties of the beam.

![Diagram of the Fiber Beam with Elastic Section](image2)

Figure 405.71: Diagram of the Fiber Beam with Elastic Section.
405.10 Statics, Elastic-Plastic, Fiber Cross Section Beam Finite Element Examples

405.10.1 Statics, Elastic-Plastic, Normal Loading and Pure Bending Fiber Cross Section Beam Finite Element

405.10.1.1 Elastic-Plastic Normal Loading, (Fiber Cross Section) Beam Finite Element Examples

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The Elastic-Plastic beam is represented by the fiber section. This example is under the load of normal loading.

![Figure 405.72: Normal Loading on the Fiber Beam with Elastic-Plastic Section.](image1)

The fiber represents the rebar. The section of all fibers represents the cross section properties of the inelastic beam.

![Figure 405.73: Diagram of the Fiber Beam with Elastic-Plastic Section.](image2)
405.10.1.2 Elastic-Plastic Pure Bending, (Fiber Cross Section) Beam Finite Element Examples

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The Elastic-Plastic beam is represented by the fiber section. This example is under the load of normal loading.

![Diagram of Fiber Beam with Elastic-Plastic Section](image1.png)

Figure 405.74: Bending on the Fiber Beam with Elastic-Plastic Section.

The fiber represents the rebar. The section of all fibers represents the cross section properties of the inelastic beam.

![Diagram of Fiber Beam with Elastic-Plastic Section](image2.png)

Figure 405.75: Diagram of the Fiber Beam with Elastic-Plastic Section.
405.11 Statics, Elastic, Inelastic Wall Finite Element Examples

405.11.1 Statics, Linear Elastic, Wall Finite Element Examples

405.11.1.1 Statics, Linear Elastic, Wall Finite Element Examples
405.11.1.2 Linear Elastic, Bi-Axial, Wall Finite Element Examples
405.11.3 Linear Elastic, Shear, (Fiber Cross Section) Wall Finite Element Examples
405.12 Statics, Elastic-Plastic Wall Finite Element Examples

405.12.1 Statics, Elastic-Plastic, in Plane, Wall Finite Element Examples

405.12.1.1 Elastic-Plastic, Uni-Axial, Wall Finite Element Examples
405.12.1.2 Elastic-Plastic, Bi-Axial, Wall Finite Element Examples
405.12.3 Elastic-Plastic, Shear, Wall Finite Element Examples
405.13 Statics, Solution Advancement Control

405.13.1 Increments: Load Control

When load-control is used as the solution advancement method, perfectly plastic model will fail immediately after the yield point. Load-control works with isotropic hardening and kinematic hardening.

405.13.1.1 Solids Example, Elastic Plastic Isotropic Hardening

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

![Model Description Diagram]

Figure 405.76: Pure Shear Cyclic Loading.

Material Response at Gauss Point:

![Material Response Graph]

Figure 405.77: Material von-Mises Isotropic Hardening under Pure Shear Cyclic Loading.
405.13.1.2 Solids Example, Elastic Plastic Kinematic Hardening

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Model Description:

![Pure Shear Cyclic Loading](image1)

Figure 405.78: Pure Shear Cyclic Loading.

Material Response at Gauss Point:

![Material von-Mises Kinematic Hardening](image2)

Figure 405.79: Material von-Mises Kinematic Hardening under Pure Shear Cyclic Loading.
405.13.1.3 Inelastic Beam Example, Steel and Reinforced Concrete

The Real-ESSI input files for this example are available HERE.

The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The Elastic-Plastic beam is represented by the fiber section. This example is under the load of normal loading.

![Normal Loading on the Beam with Fiber Section.](image)

The fiber represents the rebar. The section of all fibers represents the cross section properties of the inelastic beam.

![Diagram of the Beam with Fiber Section.](image)

405.13.2 Statics, Increments: Displacement Control

405.13.2.1 Statics, Single Displacement Control

405.13.2.2 Solids Example, Elastic-Perfectly Plastic

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Material Response at Gauss Point:
405.13.2.3 Solids Example, Elastic Plastic Isotropic Hardening

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Model Description:

Material Response at Gauss Point:

405.13.2.4 Solids Example, Elastic Plastic Kinematic Hardening

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Model Description:

Material Response at Gauss Point:
Figure 405.84: Pure Shear Cyclic Loading.

Figure 405.85: Displacement-Control of Isotropic Hardening Material under Pure Shear Cyclic Loading.

Figure 405.86: Pure Shear Cyclic Loading.
405.13.2.5 Inelastic Beam Example, Steel and Reinforced Concrete

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The Elastic-Plastic beam is represented by the fiber section. This example is under the load of normal loading.

The fiber represents the rebar. The section of all fibers represents the cross section properties of the inelastic beam.
Figure 405.89: Diagram of the Fiber Beam with Elastic-Plastic Section.
405.13.3 Statics, Solution Algorithms

405.13.3.1 Statics, Solution Algorithm: No Convergence Check

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

When no convergence check is used, the stress-strain curves drift away a little. The stress-strain curve did not close, as shown in Figure 405.57.

Model Description:

![Pure Shear Cyclic Loading](image)

Figure 405.90: Pure Shear Cyclic Loading.

Material Response at Gauss Point:

![Stress-Strain](image)

Figure 405.91: Results of No-Convergence-Check Pure Shear Cyclic Loading.
405.13.3.2 Statics, Solution Algorithm: Newton Algorithm

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

Model Description:

![Figure 405.92: Pure Shear Cyclic Loading.](#)

Material Response at Gauss Point:

![Figure 405.93: Results of Convergence Check with Newton-Raphson Iteration under Pure Shear Cyclic Loading.](#)
405.13.3 Statics, Solution Algorithm: Newton Algorithm with Line Search

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

![Diagram of Pure Shear Cyclic Loading](image)

Figure 405.94: Pure Shear Cyclic Loading.

Material Response at Gauss Point:

![Stress-Strain Material Behavior](image)

Figure 405.95: Results of Convergence Check with Newton-Raphson Iterations and Line Search under Pure Shear Cyclic Loading.
405.13.4 Statics, Solution Advancement Control, Iterations: Convergence Criteria

405.13.4.1 Statics, Solution Advancement Control, Convergence Criteria: Unbalanced Force

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

![Model Description Image]

Figure 405.96: Pure Shear Cyclic Loading.

Material Response at Gauss Point:

![Material Response Image]

Figure 405.97: Results of Convergence Check with Unbalanced Force Criteria under Pure Shear Cyclic Loading.
405.13.4.2 Statics, Solution Advancement Control, Convergence Criteria: Displacement Increment

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model Description:

Figure 405.98: Pure Shear Cyclic Loading.

Material Response at Gauss Point:

Figure 405.99: Results of Convergence Check with Displacement Increment under Pure Shear Cyclic Loading.
405.13.5 Statics, Solution Advancement Control, Different Convergence Tolerances

(Examples in preparation)
405.14 Statics, Small Practical Examples

405.14.1 Statics, Elastic Beam Element for a Simple Frame Structure

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

405.14.1.1 Problem Description

- Dimensions: width=6m, height=6m, force=100N

- Element dimensions: length=6m, cross section width=1m, cross section height=1m, mass density $\rho = 0.0\text{kN/m}^3$, Young’s modulus $E = 1\times10^8\text{ Pa}$, Poisson’s ratio $\nu = 0.0$.

![Elastic frame with beam_elastic elements.](image)

Figure 405.100: Elastic frame with beam_elastic elements.
405.14.2 Statics, 4NodeANDES Square Plate, Four Edges Clamped

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

405.14.2.1 Problem description:

Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are **clamped**.

The load is a self weight.

![Figure 405.101: Square plate with four edges clamped.](image)

405.14.2.2 Numerical model:

The element side length is 1 meter.
Figure 405.102: 4NodeANDES edge clamped square plate with element side length 1m.
405.14.3 Statics, Six Solid Blocks Example With Interface/Contact

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

This is a 3-D solid block example with initial normal and then tangential load on different surfaces as shown below.

![Illustration of Six Solid Blocks Example with Interface/Contact with first normal and then tangential loading stages.](image)

Figure 405.103: Illustration of Six Solid Blocks Example with Interface/Contact with first normal and then tangential loading stages.

The generalized displacement field of the two loading stages normal loading and tangential loading is shown below.

![Generalized displacement magnitude visualization of normal loading.](image)

Figure 405.104: Generalized displacement magnitude visualization of normal loading.
Figure 405.105: Generalized displacement magnitude visualization of tangential loading.
Chapter 406

Dynamic Examples


(In collaboration with Prof. José Abell, Dr. Yuan Feng, Mr. Sumeet Kumar Sinha, Dr. Hexiang Wang, and Dr. Han Yang)
406.1 Chapter Summary and Highlights

In this Chapter dynamic/transient modeling and simulation of solids and structures is illustrated through a number of examples.

All the examples described here, and many more, organized in sub-directories, for constitutive behavior, static and dynamic behavior can be directly downloaded from a repository at: http://sokocalo.engr.ucdavis.edu/~jeremic/lecture_notes_online_material/Real-ESSI_Examples/education_examples. These examples can then be tried, analyzed using Real-ESSI Simulator that is available on Amazon Web Services (AWS) computers around the word. Login to AWS market place and search for Real-ESSI...

406.2 Dynamic Solution Advancement (in Time)

406.2.1 Dynamics: Newmark Method

406.2.1.1 Model Description

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Firstly, the model is given an initial displacement at one side. Second, the model starts free vibration.

![Figure 406.1: Problem Description for Newmark Method](image)

406.2.1.2 Results

With damping, the displacement peak is smaller and smaller. The displacement at the top is
406.2.2 Dynamics: Hilber-Hughes-Taylor ($\alpha$) Method

406.2.2.1 Model Description

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Firstly, the model is given an initial displacement at one side. Second, the model starts free vibration.

406.2.2.2 Results

With NO damping, the displacement peak keeps the same. The displacement at the top is
406.3 Dynamics: Solution Advancement: Time Step Size

406.3.1 Dynamics: Solution Advancement: Equal Time Step

406.3.1.1 Model Description

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The model is given an earthquake input motion at the bottom with equal time step. After the wave propagation, the motion at the top is recorded.

406.3.1.2 Results

The input motion is on the left, while the output motion is on the right.
406.3.2 Dynamics Solution Advancement: Variable Time Step

406.3.2.1 Model Description

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The model is given an earthquake input motion at the bottom with variable time step. After the wave propagation, the motion at the top is recorded.

![Figure 406.7: Problem Description for Newmark Method](image)

406.3.2.2 Results

The input motion is on the left, while the output motion is on the right. The input motion is in variable time step. As shown in Fig 406.8, from time 10-11 second, the input motion is a straight line (a big time step) without the small time steps.
406.4 Dynamics: Energy Dissipation, Damping

406.4.1 Dynamics: Energy Dissipation: Viscous Damping

406.4.1.1 Dynamics: Energy Dissipation, Viscous Damping: Rayleigh Damping

Model Description The Real-ESSI input files for this example are available [here](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [here](#).

Firstly, the model is given an initial displacement at the top from 0 to 1 second. Second, after the time 1 second, the model starts free vibration.

![Figure 406.9: Problem Description for Newmark Method](#)

Results This model employs Rayleigh damping. The displacement at the top is

406.4.1.2 Dynamics: Energy Dissipation, Viscous Damping: Caughey Damping

Model Description The Real-ESSI input files for this example are available [here](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [here](#).

Firstly, the model is given an initial displacement at the top from 0 to 1 second. Second, after the time 1 second, the model starts free vibration.
**Results**  This model employs Caughey damping. The displacement at the top is
406.4.2  Dynamics: Energy Dissipation: Material (Elastic-Plastic, Hysteretic) Damping

406.4.2.1  Dynamics: Energy Dissipation, Material Damping: Elastic Perfectly Plastic Models

Model Description  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The model is a one-element solid brick example with perfectly plastic materials.

![Figure 406.13: Problem Description for Newmark Method](image)

Results  The Hysteretic loop at the Gauss point is

![Figure 406.14: Results for Newmark Method](image)
406.4.2.2 Dynamics: Energy Dissipation, Material/Hysteretic Damping: Elastic Plastic Isotropic Hardening Models

**Model Description**  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The model is a one-element solid brick example with isotropic hardening materials.

![Figure 406.15: Problem Description for Newmark Method](image)

**Results**  The Hysteretic loop at the Gauss point is

![Figure 406.16: Results for Newmark Method](image)

406.4.2.3 Dynamics: Energy Dissipation, Material/Hysteretic Damping: Elastic Plastic Kinematic Hardening Models

**Model Description**  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.
The model is a one-element solid brick example with kinematic hardening materials.

Results

The Hysteretic loop at the Gauss point is

![Hysteretic loop diagram](image)

Figure 406.17: Problem Description for Newmark Method

406.4.2.4 Dynamics: Energy Dissipation, Material/Hysteretic Damping: Elastic Plastic Armstrong-Frederick Models

Model Description

The Real-ESSI input files for this example are available [HERE](#). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

The model is a one-element solid brick example with materials with nonlinear hardening Armstrong-Frederick.

Results

The Hysteretic loop at the Gauss point is
406.4.3 Dynamics: Energy Dissipation: Numerical Damping

406.4.3.1 Energy Dissipation, Numerical Damping: Newmark Method

Model Description  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Firstly, the model is given an initial displacement in the first loading stage. In the second loading stage, the model starts free vibration.

Results  This model employs Newmark numerical damping. The displacement at the top in the second loading stage is
406.4.3.2 Dynamics: Energy Dissipation, Numerical Damping: Hilber-Hughes-Taylor (α) Method

Model Description  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Firstly, the model is given an initial displacement in the first loading stage. In the second loading stage, the model starts free vibration.

Results  This model employs HHT numerical damping. The displacement at the top in the second loading stage is
406.5 Dynamics: Elastic Solid Dynamic Examples

406.5.1 Model Description

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Firstly, the model is given an initial displacement at the top from 0 to 1 second. Second, after the time 1 second, the model starts free vibration.

406.5.2 Results

This model employs Caughey damping. The displacement at the top is
406.6 Dynamics: Elastic Structural Dynamic Examples

406.6.1 Model Description

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Firstly, the model is given an initial displacement in the first loading stage. In the second loading stage, the model starts free vibration.

406.6.2 Results

With NO damping, the displacement peak keeps the same. The displacement at the top is
Figure 406.27: Problem Description for Newmark Method

Figure 406.28: Results for Newmark Method
406.7 Dynamics: Interface/Contact Elements

406.7.1 Dynamics: Hard Interface/Contact: One Bar Normal Interface/Contact Dynamics

406.7.1.1 Model Description

This is an example of a ball, bouncing on a solid flat surface. There is only normal contact/interface between the ball and the floor. An upward force is first applied to the concentrated mass lifting it up by $0.1m$ and then the force is removed, resulting in free vibration of the ball. An illustrative diagram of the problem is shown below.

![Illustration of one bar normal contact/interface dynamics.](image)

The same example can be modeled with different contact/interface and simulation parameters as shown below. For all the different cases shown below, no numerical damping is applied. Only the contact parameters are changed to expose their functionality. The response of node 2 is plotted for all the cases.
406.7.1.2 Dynamics: No Viscous Damping

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Results Here, no viscous damping between the contact/interface pair nodes is applied. The displacement output of Node 2 is shown below.

![Displacement of Node 2](image)

Figure 406.30: Displacement of Node 2

406.7.1.3 Dynamics: Normal Viscous Damping Between Interface/Contact Node Pairs

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Results Viscous damping between the contact/interface pair nodes is applied in normal contact/interface direction. The displacement output of Node 2 is shown below.

![Displacement of Node 2](image)

Figure 406.31: Displacement of Node 2
406.7.1.4 Dynamics: Explicit Simulation

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

**Results**  With no viscous damping, the analysis is run explicitly without any convergence check. The displacement output of Node 2 is shown below.

![Displacement of Node 2](image)

Figure 406.32: Displacement of Node 2
406.7.2 Dynamics: Hard Interface/Contact: Frictional Single Degree of Freedom Problem

Model Description This is an example of a block on a rough surface under gravity. It has been attached to a spring at one end. At the other end a tangential load is applied greater than the coulomb friction and is then removed. The block oscillates back and forth with continuously losing energy because of frictional force and then stops, with some permanent deformation. This kind of damping is called frictional damping which is linear as compared to exponential in case of viscous damping. An illustrative diagram of the problem is shown below.

The same example can be modeled with different contact/interface and simulation parameters as shown below. For all the different cases shown below, no numerical damping is applied. Only the contact/interface parameters are changed to expose their functionality. The response of node 2 is plotted for all the cases.
406.7.2.1 Dynamics: No Viscous Damping

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

**Results** In this examples, no viscous damping between the contact/interface pair nodes is applied. The displacement output of *Node 2* is shown below.

![Displacement of Node 2](image1.png)

Figure 406.34: Displacement of Node 2

406.7.2.2 Dynamics: Tangential Viscous Damping Between Interface/Contact Node Pairs

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

**Results** Viscous damping between the contact/interface pair nodes is applied in tangential contact/interface direction. The displacement output of *Node 2* is shown below.

![Displacement of Node 2](image2.png)

Figure 406.35: Displacement of Node 2
406.7.2.3 Dynamics: Explicit Simulation

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Results  With no viscous damping, the analysis is run explicitly without any convergence check. The displacement output of Node 2 is shown below.

![Displacement of Node 2](image)

Figure 406.36: Displacement of Node 2
406.7.3 Dynamics: Soft Interface/Contact: One Bar Normal Interface/Contact Dynamics

Model Description  This is an example of a ball, bouncing on a solid flat surface. There is only normal contact/interface between the ball and the floor. An upward force is first applied to the concentrated mass lifting it up by $0.1m$ and then the force is removed, resulting in free vibration of the ball. An illustrative diagram of the problem is shown below.

![Illustration of one bar normal contact/interface dynamics](image)

Figure 406.37: Illustration of one bar normal contact/interface dynamics

The same example can be modeled with different contact/interface and simulation parameters as shown below. For all the different cases shown below, no numerical damping is applied. Only the contact/interface parameters are changed to expose their functionality. The response of node 2 is plotted for all the cases.
406.7.3.1 Dynamics: No Viscous Damping

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Results  In this example, no viscous damping between the contact/interface pair nodes is applied. The displacement output of Node 2 is shown below.

![Displacement of Node 2](image1)

Figure 406.38: Displacement of Node 2

406.7.3.2 Dynamics: With Normal Viscous Damping Between Interface/Contact Node Pairs

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Results  Viscous damping between the contact/interface pair nodes is applied in normal contact/interface direction. The displacement output of Node 2 is shown below.

![Displacement of Node 2](image2)

Figure 406.39: Displacement of Node 2
406.7.3.3 Dynamics: Explicit Simulation

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Results With no viscous damping, the analysis is run explicitly without any convergence check. The displacement output of Node 2 is shown below.

![Displacement of Node 2](image)

Figure 406.40: Displacement of Node 2
406.7.4 Dynamics: Soft Interface/Contact: Frictional Single Degree of Freedom Problem

**Model Description**  
This is an example of a block on a rough surface under gravity. It has been attached to a spring at one end. At the other end a tangential load is applied greater than the coulomb friction and is then removed. The block oscillates back and forth with continuously loosing energy because of frictional force and then stops, with some permanent deformation. This kind of damping is called frictional damping which is linear as compared to exponential in case of viscous damping. An illustrative diagram of the problem is shown below.

![Illustration of frictional single degree of freedom problem](attachment:frictional_single_degree_of_freedom.png)

The same example can be modeled with different contact/interface and simulation parameters as shown below. For all the different cases shown below, no numerical damping is applied. Only the contact/interface parameters are changed to expose their functionality. The response of node 2 is plotted for all the cases.
406.7.4.1 Dynamics: No Viscous Damping

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

**Results** In this example, no viscous damping between the contact/interface pair nodes is applied. The displacement output of *Node 2* is shown below.

![Graph showing displacement of Node 2](image)

Figure 406.42: Displacement of Node 2

406.7.4.2 Dynamics: Tangential Viscous Damping Between Interface/Contact Node Pairs

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

**Results** Viscous damping between the contact/interface pair nodes is applied in tangential contact/interface direction. The displacement output of *Node 2* is shown below.

![Graph showing displacement of Node 2](image)

Figure 406.43: Displacement of Node 2
406.7.4.3 Dynamics: Explicit Simulation

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Results With no viscous damping, the analysis is run explicitly without any convergence check. The displacement output of Node 2 is shown below.

![Figure 406.44: Displacement of Node 2](image-url)
406.7.5 Dynamics: Split Beam

406.7.5.1 Model Description

In this example, a normal beam is split into two halves along its depth. A uniform surface load of 50 Pa is applied to the top half of the beam, pulling it away from its lower part. Then, the load is removed, to allow free vibration between the split beams. An illustrative diagram of the problem is shown below.

![Illustration of Split Beam Analysis](image)

Figure 406.45: Illustration of Split Beam Analysis

The same example was modelled with soft and hard contact/interface. Numerical as well as viscous damping between contact/interface pair nodes was applied. The displacement response of the extreme right mid node of top half beam is plotted.

406.7.5.2 Dynamics: Split Beam With Hard Interface/Contact

The Real-ESSI input files for this example are available [HERE](url). The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](url).

![Displacement response of extreme mid node of top half beam](image)

Figure 406.46: Displacement response of extreme mid node of top half beam
406.7.5.3 Dynamics: Split Beam With Soft Interface/Contact

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

![Graph showing displacement response of extreme mid node of top half beam]

Figure 406.47: Displacement response of extreme mid node of top half beam
406.7.6 Dynamics: Block on Soil ESSI

Model Description  The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

A solid block is placed in the soil. There is contact/interface between the interface of solid and the soil. First, self-weight and then a uniform acceleration in x-direction is applied to the whole model. This analysis would provide relative displacement, velocity and acceleration response for the given shaking. An illustrative diagram of the problem is shown below.

![Illustration of frictional single degree of freedom problem](image)

Figure 406.48: Illustration of frictional single degree of freedom problem

![Applied Motion](image)

Figure 406.49: Applied Motion

Results  Displacement response of the top of the solid block is shown below. Numerical Damping, Raleigh damming and viscous damping between contact/interface node pairs are applied.
Figure 406.50: Displacement response at the top of the block

Figure 406.51: Simulation results visualization
406.8 Dynamics: Inelastic Solid Examples

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Firstly, the model is given an initial displacement at the top from 0 to 1 second. Second, after the time 1 second, the model starts free vibration.

![Problem Description for Newmark Method](image)

**Results**  This model has material damping. The displacement at the top is

![Results for Newmark Method](image)
406.9 Dynamics: Inelastic Structural Examples

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

The column beam is represented by the fiber section. This example is under the dynamic load of ground motion.

![Ground Motion on the Fiber Beam with Column Section](image)

Figure 406.54: Ground motion on the Fiber Beam with Column Section

The fiber represents the rebar. The section of all fibers represents the cross section properties of the inelastic beam.

![Diagram of the Fiber Beam with Column Section](image)

Figure 406.55: Diagram of the Fiber Beam with Column Section
406.10 Dynamics: Domain Reduction Method (DRM)

406.10.1 Dynamics: DRM One Dimensional (1D) Model

The Real-ESSI input files with 8NodeBrick for this example are available
HERE.

The same model for this example with 27NodeBrick is available
HERE.

A simple 1D DRM model is shown in Fig.(707.44). The “DRM element”, ”Exterior node” and
”Boundary node” are required to be designated in the DRM HDF5 input. The format and script for the
HDF5 input is available in DSL/input manual.

![Figure 406.56: 1D DRM model.](image)

Numerical model

Long 1D DRM model 1000:1   The Real-ESSI input files for this example are available
HERE.

The results can also be seen from this
ANIMATION.

To show the wave propagation explicitly, a long 1D model (1000:1) similar to the 1D DRM model
above was made in this section.

The model description is same to Fig.(707.44) except this model use far more soil elements.
The general view is shown in Fig.(707.46) below.

There is still now outgoing waves at the exterior layers, which is shown in Fig(707.47).
Figure 406.57: 1D DRM model.
Figure 406.58: Long 1D DRM model
Figure 406.59: Long 1D DRM model: exterior layer
406.10.2 Dynamics: Three Dimensional (3D) DRM Model

**Problem description** The Real-ESSI input files with 8NodeBrick for this example are available [HERE](#).

The same model for this example with 27NodeBrick is available [HERE](#).

As shown in Fig.(707.48), the DRM layer is used to add the earthquake motion.

![Diagram of 3D Domain Reduction Method example](image)

Figure 406.60: The diagram for 3D Domain Reduction Method example.

**Numerical result**
Figure 406.61: Diagram for the 3D DRM model.
406.10.3 Dynamics: DRM Model with Structure

Problem description  The Real-ESSI input files for this example are available [HERE](#).

The compressed package of Real-ESSI input files and postprocessing results for this example is available [HERE](#).

As shown in Fig.(406.62), the structure is placed in the middle. Five different materials are assigned to structure, contact/interface zones, soil, DRM layer, and damping layers, respectively.

![Figure 406.62: A Domain Reduction Method example with a Simple Structure.](#)
406.11 Dynamics: Eigen Analysis

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files and postprocessing results for this example is available HERE.

Model is a brick beam with distributed mass.

![Problem Description for Newmark Method](image)

**Figure 406.63: Problem Description for Newmark Method**

![Solid Brick Cantilever Eigen Mode 1, 3, 4](image)

**Figure 406.64: Solid Brick Cantilever Eigen Mode 1, 3, 4(From left to Right)**

Results
406.12  Dynamics: Fully Coupled u-p-U Element  (example in development)
406.13 Dynamics: Partially Saturated / Unsaturated u-p-U Element

(example in development)
406.14  Dynamics: Fully Coupled u-p Element (example in development)
406.15  Dynamics: Coupled Interface/Contact Element  (example in development)
406.16  Dynamics: Buoyant Forces  (example in development)
Chapter 407

Stochastic Examples

(2018-2019-2020-2021-)

(In collaboration with Dr. Hexiang Wang)
407.1 Chapter Summary and Highlights

In this Chapter stochastic/probabilistic modeling and simulation is illustrated through a number of examples. These examples can then be analyzed using Real-ESSI Simulator that is available either as a Docker container, and on Amazon Web Services (AWS) computers. Please refer to the Real-ESSI web site [real-essi.us](http://real-essi.us), for more information on how to install Real-ESSI on your computer (Linux, Windows, MacOS...).

407.2 Probabilistic Constitutive Modeling

407.2.1 Probabilistic Constitutive Modeling: Linear Elastic

The model description:

The Real-ESSI input files for this example are available in a zip archive [HERE](https://example.com).

A stochastic uniaxial elastic material with lognormal distributed random elastic modulus, mean 155 MPa and coefficient of variation 30%.

Results:

The probabilistic stress strain response of the stochastic uniaxial elastic material is shown in Figure 407.1.

![Figure 407.1: Constitutive behavior of stochastic uniaxial elastic material.](image-url)
407.2.2 Probabilistic Constitutive Modeling: Elasto-Plastic

The model description:

The Real-ESSI input files for this example are available in a zip archive [HERE](#).

A stochastic uniaxial elastoplastic material with vanishing elastic region and nonlinear Armstrong-Frederick kinematic hardening rule is modeled. The model parameters are: Armstrong-Frederick parameter $H_a$ follows lognormal distribution with marginal mean 12 MPa and coefficient of variation 20%. Armstrong-Frederick parameter $C_r$ follows lognormal distribution with marginal mean 200 and coefficient of variation (CV) 20%.

Results:

The probabilistic stress strain response of the stochastic uniaxial elastic material is shown in Figure 407.2.

![Figure 407.2: Constitutive behavior of stochastic uniaxial elastoplastic material.](#)

407.3 Probabilistic Characterization of Seismic Motions

The model description:

The Real-ESSI input files for this example are available in a zip archive [HERE](#).

For stochastic analysis with uncertain seismic excitations, it is important to characterize input uncertain motions as a non-stationary random process. The random process can be quantified through marginal mean, marginal standard deviation and correlation structure, and can be represented as Her-
mite polynomial chaos (PC). This example presents such a random process of seismic motions with marginal mean, marginal standard deviation and correlation structure defined through plain text files. It is noted that this random process is used as input bedrock excitations in the subsequent stochastic wave propagation analysis.

Results:

It is important to check that the statistics synthesized from PC representation matches well with the input. Figures 407.3 and 407.3 compare the marginal statistics and correlation structure synthesized from PC representation with the target input.

![Marginal Statistics Comparison](image)

Figure 407.3: Verification of marginal statistics of random process motions.

### 407.4 1D Stochastic Seismic Wave Propagation

#### 407.4.1 1D Stochastic Seismic Wave Propagation: Linear Elastic

The model description:

The Real-ESSI input files for this example are available in a zip archive [HERE](#).

Presented is 1D stochastic seismic wave propagation through uncertain linear elastic, layered ground. The uncertain motions characterized in section 407.3 is adopted as bedrock input. The ground is 10m
thick with three layers and discretized with 10 stochastic shear beam elements as shown in Figure 407.5.

- Layer #1: Thickness 3m, uncertain elastic modulus follows lognormal distribution with marginal mean 120 MPa and 20% coefficient of variation.
- Layer #2: Thickness 3m, uncertain elastic modulus follows lognormal distribution with marginal mean 150 MPa and 25% coefficient of variation.
- Layer #3: Thickness 4m, uncertain elastic modulus follows lognormal distribution with marginal mean 180 MPa and 25% coefficient of variation.

The correlation structure of the uncertain elastic modulus random field follows exponential correlation with correlation length as 10m.

![Figure 407.4: Input correlation structure (Left) and PC-synthesized correlation structure (Right).](image)

![Figure 407.5: 1D layered ground and stochastic shear beam FEM model.](image)
Results:

Time evolving marginal mean and marginal standard deviation of surface probabilistic displacement and acceleration response are shown in Figure 407.6 and 407.7.

Figure 407.6: Probabilistic displacement response of ground surface.

Figure 407.7: Probabilistic acceleration response of ground surface.
407.4.2 1D Stochastic Seismic Wave Propagation: Elasto-Plastic

The model description:

The Real-ESSI input files for this example are available in a zip archive HERE.

The model geometry and input seismic excitations are identical to the example in section 407.4.1. The only difference is the constitutive model of soil. In this example, probabilistic elastoplastic soil model with vanishing elastic region and Armstrong-Frederick kinematic hardening is adopted.

Results:

Time evolving marginal mean and marginal standard deviation of surface probabilistic displacement and acceleration response are shown in Figure 407.8 and 407.9.

![Figure 407.8: Probabilistic displacement response of ground surface.](image)

407.5 1D Stochastic Seismic Wave Propagation: Sobol Sensitivity Analysis

The Real-ESSI input files for this example are available in a zip archive HERE.

Sobol sensitivity analysis is performed for the stochastic wave propagation example in section 407.4.1.
From the sensitivity analysis results for probabilistic response at ground surface, it is shown that for this specific case most of the variance comes from the uncertain input motions.
Chapter 408

Three Day Short Course Examples

(In collaboration with Dr. Yuan Feng and Dr. Han Yang)
408.1 Nonlinear Analysis Steps

408.1.1 Free Field 1C

Elastic Material. The Real-ESSI input files for elastic example are available HERE.

The modeling parameters are listed below:

- Elastic Material Properties
  - Mass Density, \( \rho \), 2000 kg/m\(^3\)
  - Shear wave velocity, \( V_s \), 500 m/s
  - Young's modulus, \( E \), 1.1 GPa
  - Poisson's ratio, \( \nu \), 0.1

Elastoplastic Material, von Mises with Armstrong-Frederick Kinematic Hardening The Real-ESSI input files for elastoplastic material example are available HERE.

The modeling parameters are listed below:

- von-Mises nonlinear hardening material model
  - Mass density, \( \rho \), 2000 kg/m\(^3\)
  - Shear wave velocity, \( V_s \), 500 m/s
  - Young's modulus, \( E \), 1.1 GPa
  - Poisson's ratio, \( \nu \), 0.1
  - von Mises radius, \( k \), 60 kPa
  - Nonlinear kinematic hardening, \( H_a \), 30 MPa
  - Nonlinear kinematic hardening, \( C_r \), 60
  - Shear strength (\( \approx \sqrt{2/3 \ H_a/C_r} \)), \( S_u \), 408 kPa
  - Isotropic hardening rate, \( K_{iso} \), 0 Pa

Results of the simulation are shown in Fig. 408.1.
The time series of simulation results is shown in Fig. 408.3.
The response spectrum of motion is shown in Fig. 408.4.
Figure 408.1: Simulation model.

Figure 408.2: Simulation model.
Figure 408.3: Simulation results: acceleration time series.
Figure 408.4: Simulation results: response spectrum at soil top.
408.1.2 Free Field 3C

**Elastic Material.** The compressed package of input files for this example is HERE.

The Modeling parameters are listed below:

- Elastic Material Properties
  - Mass density, \( \rho \), 2000 kg/m\(^3\)
  - Shear wave velocity, \( V_s \), 500 m/s
  - Young's modulus, \( E \), 1.1 GPa
  - Poisson's ratio, \( \nu \), 0.1

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 5 minutes.

**von-Mises Armstrong-Frederick Material.** The compressed package of input files is HERE.

The Modeling parameters are listed below:

- von-Mises nonlinear hardening material model
  - Mass density, \( \rho \), 2000 kg/m\(^3\)
  - Shear wave velocity, \( V_s \), 500 m/s
  - Young's modulus, \( E \), 1.1 GPa
  - Poisson's ratio, \( \nu \), 0.1
  - von Mises radius, \( k \), 60 kPa
  - Nonlinear kinematic hardening, \( H_a \), 30 MPa
  - Nonlinear kinematic hardening, \( C_r \), 60
  - Shear strength (\( \approx \sqrt{2/3 \ H_a/C_r} \)), \( S_u \), 408 kPa
  - Isotropic hardening rate, \( K_{iso} \), 0 Pa

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 17 minutes.

**von-Mises G/Gmax Material.** The compressed package of input files is HERE.

The Modeling parameters are listed below:

- von-Mises G/Gmax material model
– Mass density, \( \rho \), 2000 \( \text{kg/m}^3 \)
– Shear wave velocity, \( V_s \), 500 \( \text{m/s} \)
– Young’s modulus, \( E \), 1.1 GPa
– Poisson’s ratio, \( \nu \), 0.1
– Total number of shear modulus 9
– \( G \) over \( G_{\text{max}} \), 1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063
– Shear strain gamma, 0, 1E-6, 1E-5, 5E-5, 1E-4, 0.0005, 0.001, 0.005, 0.01

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 565 minutes.

**Drucker-Prager G/Gmax Material.** The compressed package of input files is [HERE](#).

The Modeling parameters are listed below:

- Drucker-Prager G/Gmax material model
  - Mass density, \( \rho \), 2000 \( \text{kg/m}^3 \)
  - Shear wave velocity, \( V_s \), 500 \( \text{m/s} \)
  - Young’s modulus, \( E \), 1.1 GPa
  - Poisson’s ratio, \( \nu \), 0.1
  - Initial confining stress, \( p_0 \), 100 kPa
  - Reference pressure, \( p_{\text{refer}} \), 100 kPa
  - Pressure exponential, \( n \), 0.5
  - Cohesion, \( n \), 1 kPa
  - Total number of Shear Modulus 9
  - \( G \) over \( G_{\text{max}} \), 1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063
  - Shear strain gamma, 0, 1E-6, 1E-5, 5E-5, 1E-4, 0.0005, 0.001, 0.005, 0.01

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 565 minutes.

Results are shown in Fig. 408.56.

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 871 minutes.

The time series of simulation results is shown in Fig. 408.7.

The response spectrum of motion is shown in Fig. 408.8.
Figure 408.5: Simulation model.

Figure 408.6: Simulation model.

Figure 408.7: Simulation results: acceleration time series.
Figure 408.8: Simulation results: response spectrum at soil top.
408.1.3 Soil-Foundation Interaction 3D

**Elastic Material.** The compressed package of input files is [HERE](#).

The Modeling parameters are listed below:

- **Elastic Material Properties**
  - Mass density, \( \rho \), 2000 kg/m\(^3\)
  - Shear wave velocity, \( V_s \), 500 m/s
  - Young’s modulus, \( E \), 1.1 GPa
  - Poisson’s ratio, \( \nu \), 0.1

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 13 minutes.

**von-Mises Armstrong-Frederick Material.** The compressed package of input files is [HERE](#).

The Modeling parameters are listed below:

- von-Mises nonlinear hardening material model
  - Mass density, \( \rho \), 2000 kg/m\(^3\)
  - Shear wave velocity, \( V_s \), 500 m/s
  - Young’s modulus, \( E \), 1.1 GPa
  - Poisson’s ratio, \( \nu \), 0.1
  - von Mises radius, \( k \), 60 kPa
  - Nonlinear kinematic hardening, \( H_a \), 30 MPa
  - Nonlinear kinematic hardening, \( C_r \), 60
  - Shear strength (\( \approx \sqrt{2/3} \ H_a/C_r \)), \( S_u \), 408 kPa
  - Isotropic hardening rate, \( K_{iso} \), 0 Pa

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 36 minutes.

**von-Mises G/Gmax Material.** The compressed package of input files is [HERE](#).

The Modeling parameters are listed below:

- von-Mises G/Gmax material model
- Mass density, $\rho$, $2000 \text{ kg/m}^3$
- Shear wave velocity, $V_s$, $500 \text{ m/s}$
- Young's modulus, $E$, $1.1 \text{ GPa}$
- Poisson's ratio, $\nu$, 0.1
- Total number of shear modulus, 9
- $G$ over $G_{\text{max}}$, $1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063$
- Shear strain gamma, $0, 1\times10^{-6}, 1\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}, 0.0005, 0.001, 0.005, 0.01$

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 726 minutes.

Drucker-Prager G/Gmax Material. The compressed package of input files is HERE.

The Modeling parameters are listed below:

- Drucker-Prager G/Gmax material model
- Mass density, $\rho$, $2000 \text{ kg/m}^3$
- Shear wave velocity, $V_s$, $500 \text{ m/s}$
- Young's modulus, $E$, $1.1 \text{ GPa}$
- Poisson's ratio, $\nu$, 0.1
- Initial confining stress, $p_0$, $100 \text{ kPa}$
- Reference pressure, $p_{\text{refer}}$, $100 \text{ kPa}$
- Pressure exponential, $n$, 0.5
- Cohesion, $c$, 1 kPa
- Total number of Shear Modulus, 9
- $G$ over $G_{\text{max}}$, $1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063$
- Shear strain gamma, $0, 1\times10^{-6}, 1\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}, 0.0005, 0.001, 0.005, 0.01$

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 1252 minutes.
Contact/Interface/Joint Elements. The compressed package of input files is HERE.

The Modeling parameters are listed below:

- Elastic Material Properties
  - Mass density, $\rho$, $2000 \text{ kg/m}^3$
  - Shear wave velocity, $V_s$, $500 \text{ m/s}$
  - Young’s modulus, $E$, $1.1 \text{ GPa}$
  - Poisson’s ratio, $\nu$, $0.1$

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 24 minutes.

Both Elastoplastic Material and Contact/Interface/Joint Elements. The compressed package of input files is HERE.

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 41 minutes.

Results of the simulation are shown in Fig. 408.12.
Figure 408.10: Soil foundation interaction results.
408.1.4 Soil-Structure Interaction 3D

Elastic Material. The compressed package of input files is HERE.

The Modeling parameters are listed below:

- Elastic Material Properties
  - Mass density, $\rho$, 2000 $kg/m^3$
  - Shear wave velocity, $V_s$, 500 $m/s$
  - Young’s modulus, $E$, 1.1 GPa
  - Poisson’s ratio, $\nu$, 0.1

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 10 minutes.

von-Mises Armstrong-Frederick Material. The compressed package of input files is HERE.

The Modeling parameters are listed below:

- von-Mises nonlinear hardening material model
  - Mass density, $\rho$, 2000 $kg/m^3$
  - Shear wave velocity, $V_s$, 500 $m/s$
  - Young’s modulus, $E$, 1.1 GPa
  - Poisson’s ratio, $\nu$, 0.1
  - von Mises radius, $k$, 60 kPa
  - Nonlinear kinematic hardening, $H_a$, 30 MPa
  - Nonlinear kinematic hardening, $C_r$, 60
  - Shear strength ($\approx \sqrt{2/3} H_a/C_r$), $S_u$, 408 kPa
  - Isotropic hardening rate, $K_{iso}$, 0 Pa

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 46 minutes.

von-Mises G/Gmax Material. The compressed package of input files is HERE.

The Modeling parameters are listed below:

- von-Mises G/Gmax material model
- Mass density, $\rho$, 2000 kg/m$^3$
- Shear wave velocity, $V_s$, 500 m/s
- Young’s modulus, $E$, 1.1 GPa
- Poisson’s ratio, $\nu$, 0.1
- Total number of shear modulus 9
- $G$ over $G_{\text{max}}, 1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063$
- Shear strain gamma, 0, $1E^{-6}, 1E^{-5}, 5E^{-5}, 1E^{-4}, 0.0005, 0.001, 0.005, 0.01$

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 755 minutes.

**Drucker-Prager $G/G_{\text{max}}$ Material.** The compressed package of input files is [HERE](#).

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 1178 minutes.

The Modeling parameters are listed below:

- Drucker-Prager $G/G_{\text{max}}$ material model
  - Mass density, $\rho$, 2000 kg/m$^3$
  - Shear wave velocity, $V_s$, 500 m/s
  - Young’s modulus, $E$, 1.1 GPa
  - Poisson’s ratio, $\nu$, 0.1
  - Initial confining stress, $p_0$, 100 kPa
  - Reference pressure, $p_{\text{ref}}$, 100 kPa
  - Pressure exponential, $n$, 0.5
  - Cohesion, $c$, 1 kPa
  - Total number of Shear Modulus 9
  - $G$ over $G_{\text{max}}$, 1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063
  - Shear strain gamma, 0, $1E^{-6}, 1E^{-5}, 5E^{-5}, 1E^{-4}, 0.0005, 0.001, 0.005, 0.01$

**SIMULATION TIME:** With 8 cores, the running time for this example is

**Contact/Interface/Joint Elements.** The compressed package of input files is [HERE](#).

**SIMULATION TIME:** With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 15 minutes.
Both Elastoplastic Material and Contact/Interface/Joint Elements. The compressed package of input files is HERE.

The thickness of the shell structure is 2 meters.

Results of the simulation are shown in Fig. 408.12.

SIMULATION TIME: With 8 cores on AWS EC2 c4.2xlarge instance, the running time for this example is 47 minutes.

Simulation with 1C motion. The time series of simulation results is shown in Fig. 408.13.

The response spectrum of motion is shown in Fig. 408.14.

Simulation with 3 \times 1C motion. The time series of simulation results is shown in Fig. 408.15.
Figure 408.12: Simulation Model.

Figure 408.13: Simulation Results: Acceleration Time Series with 1C motion.

The response spectrum of motion is shown in Fig. 408.16.
Figure 408.14: Simulation Results: Response Spectrum of Structure Top with 1C motion.
Figure 408.15: Simulation Results: Acceleration Time Series with 3C motion.
Figure 408.16: Simulation Results: Response Spectrum of Structure Top with 3C motion.
408.1.5 Analysis of a Structure without Soil

408.1.5.1 Eigen Analysis

Eigen analysis of a fixed base structural model should provide a good check of the structural model, natural (eigen) frequencies, and natural (eigen) modes.

The compressed package of input files is HERE.

Figure 408.17: Structure on a fixed based simulation model.

For this particular example, eigen modes and frequencies are given in Figures 408.18 and 408.19

Figure 408.18: Eigen frequencies: $f_1 = 3.47\text{Hz}$  $f_2 = 3.47\text{Hz}$  $f_3 = 6.88\text{Hz}$ (eigen mode 1 to 3 from left to right).
Figure 408.19: Eigen frequencies: $f_4 = 11.50$Hz, $f_5 = 11.50$Hz, $f_6 = 12.13$Hz (eigen modes 4 to 6 from left to right).

Input files for eigen analysis of the fixed base structure are available at this LINK, and can be directly simulated using Real-ESSI Simulator, http://real-essi.us/, that is available on Amazon Web Services, https://aws.amazon.com/.
408.1.5.2 Imposed Motion

The Real-ESSI input files for this example are available [HERE](#). The compressed package of input files is [HERE](#).

In addition to eigen analysis, fixed base structural model is used to test response of a fixed base structure. This is important as it provides an opportunity to compare results between different finite element programs, some of which can only model dynamics of fixed base structures.1

The simulation model is shown below.

![Simulation Model](#)

**Figure 408.20: Simulation Model.**

The simulation results:

![Simulation Results](#)

**Figure 408.21: Simulation Results.**

The time series of simulation results is shown in Fig. 408.22.
Figure 408.22: Simulation Results: Acceleration Time Series with 1C imposed motion.

The response spectrum of motion is shown in Fig. 408.23.
Figure 408.23: Simulation Results: Response Spectrum of Structure Top with 1C imposed motion.
408.2 Day 1: Overview

408.2.1 Nuclear Power Plant with 3C motions from SW4

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Simulation Model](image)

Figure 408.24: Simulation Model.

The Modeling parameters are listed below:

- Soil
  - Unit weight, $\gamma$, 21.4 kPa
  - Shear velocity, $V_s$, 500 m/s
  - Young's modulus, $E$, 1.3 GPa
  - Poisson's ratio, $\nu$, 0.25
  - Shear strength, $S_u$, 650 kPa
  - von Mises radius, $k$, 60 kPa
  - kinematic hardening, $H_0$, 30 MPa
  - kinematic hardening, $C_r$, 25

- Structure
  - Unit weight, $\gamma$, 24 kPa
- Young’s modulus, $E$, 20 GPa
- Poisson’s ratio, $\nu$, 0.21

The input motion at the bottom is a 3C wave from SW4.

SIMULATION TIME: With 32 cores on AWS EC2 c4.8xlarge instance, the running time for this example is 17 hours.
408.2.2 Nuclear Power Plant with 1C motions from Deconvolution

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Simulation Model](image)

Figure 408.25: Simulation Model.

The input motion at the bottom is the deconvolution of the Northridge earthquake records.

![Motion Deconvolution](image)

Figure 408.26: Motion Deconvolution.

The Modeling parameters are listed below:

- Soil
  - Unit weight, $\gamma$, 21.4 kPa
  - Shear velocity, $V_s$, 500 m/s
  - Young’s modulus, $E$, 1.3 GPa
  - Poisson’s ratio, $\nu$, 0.25
- Shear strength, $S_u$, 650 kPa
- von Mises radius, $k$, 60 kPa
- kinematic hardening, $H_a$, 30 MPa
- kinematic hardening, $C_r$, 25

- **Structure**

  - Unit weight, $\gamma$, 24 kPa
  - Young’s modulus, $E$, 20 GPa
  - Poisson’s ratio, $\nu$, 0.21
408.2.3 Nuclear Power Plant with $3 \times 1C$ motions from Deconvolution

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Simulation Model](image)

Figure 408.27: Simulation Model.

The input motion at the bottom is the deconvolution of the Northridge earthquake records.

![Acceleration Deconvolution](image)

Figure 408.28: Acceleration Deconvolution, from left to right in x, y, z directions respectively.

![Displacement Deconvolution](image)

Figure 408.29: Displacement Deconvolution, from left to right in x, y, z directions respectively.
The Modeling parameters are listed below:

- **Soil**
  - Unit weight, $\gamma$, 21.4 kPa
  - Shear velocity, $V_s$, 500 m/s
  - Young's modulus, $E$, 1.3 GPa
  - Poisson's ratio, $\nu$, 0.25
  - Shear strength, $S_u$, 650 kPa
  - von Mises radius, $k$, 60 kPa
  - kinematic hardening, $H_a$, 30 MPa
  - kinematic hardening, $C_r$, 25

- **Structure**
  - Unit weight, $\gamma$, 24 kPa
  - Young's modulus, $E$, 20 GPa
  - Poisson's ratio, $\nu$, 0.21
408.2.4 Single Element Models: Illustration of the Elastic-Plastic Behavior

The compressed package of Real-ESSI input files for this example with von-Mises material model are available HERE.

The compressed package of Real-ESSI input files for this example with Drucker-Prager material model are available HERE.

The Modeling parameters are listed below:

- von-Mises linear hardening material model
  - Mass Density, ρ, 0.0 kg/m³
  - Young’s modulus, E, 20 MPa
  - Poisson’s ratio, ν, 0.0
  - von Mises radius, k, 100 kPa
  - kinematic hardening rate, $K_{kine}$, 2 MPa
  - isotropic hardening rate, $K_{iso}$, 0 Pa

- Drucker-Prager nonlinear hardening material model
  - Mass Density, ρ, 0.0 kg/m³
  - Young’s modulus, E, 20 MPa
  - Poisson’s ratio, ν, 0.0
  - Drucker-Prager, k, 0.179527
  - nonlinear kinematic hardening, $H_{a}$, 20 MPa
  - nonlinear kinematic hardening, $C_{r}$, 100
  - isotropic hardening rate, $K_{iso}$, 0 Pa
  - initial confining stress, $p_{0}$, 1 Pa

Inelastic/nonlinear material behavior is shown in Fig. 408.31.
Figure 408.30: Simulation Model of Single Element.

Figure 408.31: Inelastic/Nonlinear material behavior.
408.2.5 Pushover for Nonlinear Frame

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Model for pushover simulation and the cross section of fiber beam](image)

Figure 408.32: Model for pushover simulation and the cross section of fiber beam (concrete and reinforcement).

Result are shown in Fig. 408.33.

![Results for fiber beam pushover](image)

Figure 408.33: Results for fiber beam pushover.

The Modeling parameters are listed below:

- Uniaxial concrete
  - Compressive strength, 24 MPa
  - Strain at compressive strength, 0.001752
Figure 408.34: Boundary condition $u_x$ for fiber beam pushover.

- Crushing strength, 0.0 Pa
- Strain at compressive strength, 0.003168
- $\lambda$, 0.5
- Tensile strength, 0 Pa
- Tension softening stiffness, 0 Pa

- Uniaxial steel
  - Yield strength, 413.8 MPa
  - Young’s modulus, 200 GPa
  - Strain hardening ratio, 0.01
  - $R_0$, 18.0
  - $cR1$, 0.925
  - $cR2$, 0.15
  - $a_1$, 0.0
  - $a_2$, 55.0
  - $a_3$, 0.0
  - $a_4$, 55.0
408.2.6 Pre-Processing examples with Gmsh

408.2.6.1 Cantilever Example

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.35: Simulation Model Cantilever.

Results are shown in Fig. 408.36.

Figure 408.36: Simulation model. cantilever, results.
408.2.6.2 Brick-shell-beam Example

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Simulation Model Brick-Shell-Beam](image)

Figure 408.37: Simulation Model Brick-Shell-Beam.

Results are shown in Fig. 408.38.
Figure 408.38: Brick-Shell-Beam, Results.
408.2.6.3 DRM 2D Example

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Simulation Model DRM 2D.](image1)

Results of free field DRM 2D Model under 1C motion are shown in Fig. 408.40.

![Simulation Model DRM 2D.](image2)
408.2.6.4 DRM 3D Example

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.41: Simulation Model DRM 3D.

Results of free field DRM 3D Model under 1C motion are shown in Fig. 408.42.

Figure 408.42: Simulation Model DRM 2D.
408.2.7 Post-processing examples with ParaView

408.2.7.1 Slice Visualization

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.43: Slice Visualization with ParaView.
408.2.7.2 Stress Visualization

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.44: Stress Visualization with ParaView.
408.2.7.3 Pore Pressure Visualization with upU Element

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Image of Pore Pressure Visualization with Paraview]

Figure 408.45: Pore Pressure Visualization with Paraview.
408.2.7.4 Eigen Visualization

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.46: Eigen Mode Visualization with Paraview.
408.2.8 Check Model and Visualization of Boundary Conditions

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.47: Partition Information Visualization with Paraview.

Figure 408.48: Partition Information Visualization with Paraview.
408.2.9 Restart Simulation

408.2.9.1 Restart in the next stage

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Figure 408.49: Restart Simulation.](image)

This group of examples illustrates the restart functionality between loading stages. There are three test cases in this example. The two loading stages in the first test case is split into two test cases to show the restart feature.

- The first test case run through two loading stages.

- The second test case only run the first loading stage and saves model state at the end.

- The third test case restart the simulation from the saved model state of the second test case. Then, with the restart model state, the test case run the second loading stage only.

Results of the third test case are exactly the same to the first test case.
408.2.9.2 Restart inside the stage

For the case of lack of convergence, restart with the previous loading stage.

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

This group of examples illustrate the restart functionality inside one loading stage when the simulation cannot converge in the nonlinear analysis. The nonlinear material model, von-Mises Armstrong-Frederick, is used in all test cases.

There are three test cases in this example.

- The first test case run through the whole simulation with a relatively big tolerance of the unbalanced force.
- The second test case failed in the middle of the simulation with a relatively small tolerance of the unbalanced force. When the second test failed, the model reverted to the last commit model state and saved model state.
- The third test case load the saved model state, increased the tolerance of the unbalanced force, and added the remaining load to the model to continue the simulation.

Results of the third test case are exactly the same to the first test case.

Note that in the third test case only the remaining load should be added to the model. Whenever the new loading stage is used, the previous loading are all finished, which means that the static loading becomes constant and the dynamic loading vanishes.
408.3 Day 2: Seismic Motions

408.3.1 Deconvolution and Propagation of 1C Motions, 1D Model

Various deconvolution and propagation 1D models for one component (1C) wave propagation are provided through links below.

**Note:** Please make sure that the input acceleration record is baseline corrected and the displacement record has no permanent deformation. Otherwise, the unrealistic high frequency components can be brought into the simulation results.

- Deconvolution of Ormsby wavelet, input files are available [HERE](#).
- Deconvolution of Northridge earthquake, input files are available [HERE](#).
- Deconvolution of and DRM propagation of Ormsby wavelet, input files are available [HERE](#).
- Deconvolution of and DRM propagation of Northridge earthquake, input files are available [HERE](#).

408.3.2 Convolution and Propagation of 1C Motions, 1D Model

Various convolution and propagation 1D models for one component (1C) wave propagation are provided through links below:

**Note:** Please make sure that the input acceleration record is baseline corrected and the displacement record has no permanent deformation. Otherwise, the unrealistic high frequency components can be brought into the simulation results.

- Convolution of Ormsby wavelet, input files are available [HERE](#).
- Convolution of Northridge earthquake, input files are available [HERE](#).
- Convolution of and DRM propagation of Ormsby wavelet, input files are available [HERE](#).
- Convolution of and DRM propagation of Northridge earthquake, input files are available [HERE](#).

408.3.3 Convolution, Deconvolution and Propagation of 1C Motions, 2D Model

Various convolution, deconvolution and propagation 2D models for one component (1C) wave propagation are provided through links below.

**Note #1:** Please make sure that the input acceleration record is baseline corrected and the displacement record has no permanent deformation. Otherwise, the unrealistic high frequency components can be brought into the simulation results.
Note #2: Please make sure that you develop seismic motions by doing deconvolution and then convolution before analyzing the actual model. File run.sh in examples directory has a proper sequence of commands, that is one should first run Real-ESSI on Deconvolution_DRM_motion.fei and then, when motions are developed, analyze model.

Examples are available through links below:

- Convolution/Deconvolution of and DRM propagation of Ormsby wavelet, input files are available HERE.

- Convolution/Deconvolution of and DRM propagation of Kobe earthquake records, input files are available HERE.
408.3.3.1 ESSI 3D building model, deconvolution 1C model, shell model with DRM

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- **Elastic Soil Material Properties**
  - Mass density, \( \rho \), 2000 kg/m\(^3\)
  - Shear Wave Velocity, \( V_s \), 500 m/s
  - Young’s modulus, \( E \), 1.1 GPa
  - Poisson’s ratio, \( \nu \), 0.1

- **Elastic Structure Material Properties**
  - Mass density, \( \rho \), 2500 kg/m\(^3\)
  - Young’s modulus, \( E \), 20 GPa
  - Poisson’s ratio, \( \nu \), 0.1

![Simulation Model](image)

Figure 408.51: Simulation Model.

Results of DRM 3D shell Structure Model under 1C motion are shown in Fig. 408.52.
Figure 408.52: Simulation Model.
408.3.4 Deconvolution $3 \times 1C$ Motions

408.3.4.1 Free field 1C model, deconvolution $3 \times 1C$ motion, model with DRM

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- Elastic Material Properties
  - Mass density, $\rho$, 2000 $kg/m^3$
  - Shear Wave Velocity, $V_s$, 500 $m/s$
  - Young’s modulus, $E$, 1.1 GPa
  - Poisson’s ratio, $\nu$, 0.1

![Simulation Model](image)

Figure 408.53: Simulation Model.

Results of the simulation are shown in Fig. 408.1.
Figure 408.54: Simulation Model.
408.3.4.2 Free field 3D model, deconvolution 3×1C motion, model with DRM

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- Elastic Soil Material Properties
  - Mass density, $\rho$, 2000 kg/m$^3$
  - Shear Wave Velocity, $V_s$, 500 m/s
  - Young’s modulus, $E$, 1.1 GPa
  - Poisson’s ratio, $\nu$, 0.1
408.3.4.3 ESSI 3D building model, deconvolution $3 \times 1C$ motion, shell model with DRM

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- **Elastic Soil Material Properties**
  - Mass density, $\rho$, 2000 $kg/m^3$
  - Shear Wave Velocity, $V_s$, 500 $m/s$
  - Young’s modulus, $E$, 1.1 GPa
  - Poisson’s ratio, $\nu$, 0.1

- **Elastic Structure Material Properties**
  - Mass density, $\rho$, 2500 $kg/m^3$
  - Young’s modulus, $E$, 20 GPa
  - Poisson’s ratio, $\nu$, 0.1

Results of DRM 3D shell Structure Model under 1C motion are shown in Fig. 408.58.
Figure 408.58: Simulation Model.
408.3.5 Mesh Dependence of Wave Propagation Frequencies

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Show the mesh dependence of high frequency wave with Ormsby wavelet.

Figure 408.59: Simulation Model.

Results of mesh dependence are shown in Fig. 408.60.
Figure 408.60: Convolution Results and Mesh Dependence.
408.3.6 Application of 3C Motions from SW4

408.3.6.1 3C Seismic Motion from SW4

A 3C seismic motion field has been developed by using SW4. The characteristic parameters of the seismic motion are given below:

- Geological model: length 3km, width 3km, height 1.7km, grid size 50m, width of super grid damping layer 30m.
- Material model: Elastic material, First 1km: $V_p = 4630.76m/s$, $V_s = 2437.56m/s$, $\rho = 2600kg/m^3$. 1km ∼ 1.7km: $V_p = 6000m/s$, $V_s = 3464m/s$, $\rho = 2700kg/m^3$
- Source type: point moment source, moment seismic moment $M_{xy} = 5e^{15}N \cdot m$, moment magnitude 4.5.
- Time function: Gaussian function, with dominant frequency $2.5Hz$ and maximum frequency $6.5Hz$.

The time series displacement and acceleration response at the center of the model is shown below in figure 408.61. And figure 408.62 gives corresponding FFT response.

![Time series response of 3C motion.](image)
During the simulation of SW4, the time series motions at many ESSI nodes (basically are some pre-defined record stations) of an ESSI box \((300m \times 300m \times 100m)\) are recorded and written into SAC files. Then an transition program SW42ESSI has been developed to interpolate these motions to DRM nodes of localized ESSI model by specifying some geometric translational and rotational transformation, as shown in figure ??.

To launch SW42ESSI, following parameters are needed:

- **DRM input**: specify the name of DRM input files. This DRM file just contains the geometric information of DRM layer in ESSI model (e.g. DRM node IDs, nodal coordinates, etc).
- **SW4 motion directory**: specify the output directory of SW4, that contains SAC files.
- **origin coordinates of ESSI box \((x, y, z)\)**: the SW4 coordinates of the origin of ESSI box, i.e. the coordinates of ESSI nodes, whose station ID is \((0, 0, 0)\).
- **dimensions of ESSI Box \((length, width, height)\)**: specify the dimension \((length, width and height)\) of ESSI box.
- **spacing of ESSI nodes**: specify the grid spacing of ESSI nodes (i.e. motion recording stations)
- **interval of time steps for sampling**: specify the sampling frequency, if 1 is used here, ESSI simulation time step is the same as the simulation time step of SW4.
• reference point in ESSI model for translational transformation (x, y, z): specify the coordinate of reference point for translational transformation in ESSI model.

• reference point in SW4 model for translational transformation (x, y, z): specify the coordinate of reference point for translational transformation in SW4 model.

• conduct rotational transformation (yes/no): input yes and provide more rotational transformation parameters to enable rotational transformation. If input no, no more parameters are required.

• reference point in SW4 model for rotational transformation (x, y, z): specify the coordinate of reference point for rotational transformation in SW4 model.

• degrees of rotation along three axes (x, y, z): specify the degrees of rotation along three axes. The sign of rotation degrees follows right hand rule.

---

Figure 408.63: Illustration of transition from SW4 to Real-ESSI.

408.3.6.2 Free field 3D model, 3C motion, model with DRM

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Results of free field DRM 3D Model under 3C motion are shown in figure 408.65.
Figure 408.64: Simulation Model.

Figure 408.65: Simulation of 3D free field model under 3C seismic motion.
408.3.6.3 ESSI 3D building model, 3C motion, shell model with DRM

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.66: Simulation Model.
408.4 Day 3: Inelastic, Nonlinear Analysis

408.4.1 Single Element Models: Illustration of the Elastic-Plastic Behavior

408.4.1.1 von-Mises Perfectly Plastic Material Model.

The Real-ESSI input files for von-Mises perfectly plastic example are available [HERE](#). The compressed package of Real-ESSI input files for this example is available [HERE](#).

408.4.1.2 von-Mises Armstrong-Frederick Material Model.

The Real-ESSI input files for von-Mises Armstrong-Frederick example are available [HERE](#). The compressed package of Real-ESSI input files for this example is available [HERE](#).

The Modeling parameters are listed below:

- **Left: von-Mises linear hardening material model**
  - Mass Density, $\rho$, 0.0 $kg/m^3$
  - Young’s modulus, $E$, 20 MPa
  - Poisson’s ratio, $\nu$, 0.0
  - von Mises radius, $k$, 100 kPa
  - kinematic hardening rate, $K_{kine}$, 2 MPa
  - isotropic hardening rate, $K_{iso}$, 0 Pa

- **Right: Drucker-Prager nonlinear hardening material model**
  - Mass Density, $\rho$, 0.0 $kg/m^3$
  - Young’s modulus, $E$, 20 MPa
  - Poisson’s ratio, $\nu$, 0.0
  - Drucker-Prager, $k$, 0.179527
  - nonlinear kinematic hardening, $H_a$, 20 MPa
  - nonlinear kinematic hardening, $C_r$, 100
  - isotropic hardening rate, $K_{iso}$, 0 Pa
  - initial confining stress, $p_0$, 1 Pa

Results are shown in Fig. 408.68.
Figure 408.67: Simulation Model of Single Element.

Figure 408.68: Simulation Results of Single Element.
408.4.1.3 von-Mises G/Gmax Material Model

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- von-Mises G/Gmax material model
  - Mass density, \( \rho \), 2000 \( kg/m^3 \)
  - Young’s modulus, \( E \), 200 MPa
  - Poisson’s ratio, \( \nu \), 0.1
  - Total number of shear modulus  9
  - \( G \) over \( G_{max} \), 1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063
  - Shear strain gamma, 0, 1E-6, 5E-5, 1E-4, 0.0005, 0.001, 0.005, 0.01

Figure 408.69: Simulation Model of Single Element.
408.4.1.4 Drucker-Prager Perfectly Plastic Material Model

The Real-ESSI input files for this Drucker-Prager perfectly plastic example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

408.4.1.5 Drucker-Prager Armstrong-Frederick Non-Associated Material Model

The Real-ESSI input files for this Drucker-Prager Armstrong-Frederick example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.70: Simulation model, single element.

Results are shown in Fig. 408.71.

Figure 408.71: Simulation results for single element.
408.4.1.6 Drucker-Prager G/Gmax Non-Associated Material Model

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- Drucker-Prager G/Gmax material model
  - Mass density, $\rho$, 2000 kg/m$^3$
  - Young’s modulus, $E$, 200 MPa
  - Poisson’s ratio, $\nu$, 0.1
  - Initial confining stress, $p_0$, 100 kPa
  - Reference pressure, $p_{\text{refer}}$, 100 kPa
  - Pressure exponential, $n$, 0.5
  - Cohesion, $n$, 1 kPa
  - Total number of Shear Modulus, 9
  - $G$ over $G_{\text{max}}$, 1, 0.995, 0.966, 0.873, 0.787, 0.467, 0.320, 0.109, 0.063
  - Shear strain gamma, 0, 1E-6, 1E-5, 5E-5, 1E-4, 0.0005, 0.001, 0.005, 0.01

![Simulation Model of Single Element.](image)

Results are shown in Fig. 408.73.
Figure 408.73: Simulation Results of Single Element.
408.4.2 Wave Propagation Through Elasto-plastic Soil

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.74: Wave Propagation through elastoplastic Soils.

The displacement series at the surface are plotted in time and frequency domain.

Figure 408.75: Simulation Results of Wave Propagation.
408.4.3 Contact/Interface/Joint Examples

408.4.3.1 Axial Behavior: Stress-Based Hard Contact/Interface/Joint Example

The Real-ESSI input files for hard contact/interface example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

408.4.3.2 Axial Behavior: Stress-Based Soft Contact/Interface/Joint Example

The Real-ESSI input files for soft contact/interface example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The axial behavior of hard contact/interface and soft contact/interface is illustrated in Fig. 408.76.

Figure 408.76: Simulation results for axial behavior of (left) soft contact/interface and (right) hard contact.
408.4.3.3 Shear behavior: Stress-Based Elastic Perfectly Plastic Contact/Interface/Joint

The Real-ESSI input files for the elastic-perfectly plastic example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

408.4.3.4 Shear behavior: Stress-Based Elastic-Hardening Contact/Interface/Joint

The Real-ESSI input files for the elastic-hardening contact/interface example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

408.4.3.5 Shear behavior: Stress-Based Elastic-Hardening-Softening Contact/Interface/Joint

The Real-ESSI input files for the elastic-hardening-softening example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The shear behavior of elastic-perfectly plastic, elastic-hardening plastic, elastic and hardening and softening plastic is illustrated in Fig. 408.77.
Figure 408.77: Simulation results for shear behavior for stress based contact elements: elastic-perfectly plastic, elastic-hardening plastic, elastic, hardening and softening plastic.
408.4.3.6 Force Based Contact/Interface/Joint Example: Base Isolator

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Simulation Model](image1)

Figure 408.78: Simulation Model.

Results are shown in Fig. 408.79.

![Simulation Results](image2)

Figure 408.79: Simulation Results for Contact/Interface/Joint Examples.
408.4.4  Inelastic Frame Pushover

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- **Uniaxial concrete**
  - Compressive strength, 24 MPa
  - Strain at compressive strength, 0.001752
  - Crushing strength, 0.0 Pa
  - Strain at compressive strength, 0.003168
  - lambda, 0.5
  - Tensile strength, 0 Pa
  - Tension softening stiffness, 0 Pa

- **Uniaxial steel**
  - Yield strength, 413.8 MPa
  - Young’s modulus, 200 GPa
  - Strain hardening ratio, 0.01
  - R0, 18.0
  - cR1, 0.925
  - cR2, 0.15
  - a1, 0.0
  - a2, 55.0
  - a3, 0.0
  - a4, 55.0

Result is shown in Fig. 408.81.
Figure 408.80: Model for pushover simulation and the cross section of fiber beam (concrete and reinforcement).

Figure 408.81: Results for fiber pushover.
Figure 408.82: Boundary condition $u_x$ for a fiber beam pushover.
408.4.5 Inelastic Wall Pushover

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- Concrete Wall
  - Young’s modulus, 36.9 GPa
  - Poisson’s ratio, 0.2
  - Tensile yield strength, 5 MPa
  - Compressive yield strength, 56 MPa
  - Plastic deformation rate, 0.4
  - Damage parameter Ap, 0.1
  - Damage parameter An, 1.5
  - Damage parameter Bn, 0.75

- Uniaxial steel
  - Yield strength, 457.5 MPa
  - Young’s modulus, 200 GPa
  - Strain hardening ratio, 0.011042
  - a1, 0.0
  - a2, 55.0
  - a3, 0.0
  - a4, 55.0
Figure 408.83: Model for wall element pushover.
408.4.6 Viscous Nonlinear behavior

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

Figure 408.84: Simulation Model.

Result are shown in Fig. 408.85 and Fig. 408.86.

Figure 408.85: Results for low viscous damping.
Figure 408.86: Results for high viscous.
408.4.7 Numerical Damping Example

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

![Simulation Model](image1)

Figure 408.87: Simulation Model.

Result are shown in Fig. 408.85 and Fig. 408.89.

![Results of low numerical damping](image2)

Figure 408.88: Results of low numerical damping.
Figure 408.89: Results of high numerical damping.
408.4.8 Nuclear Power Plant Example with Nonlinearities

The Real-ESSI input files for this example are available HERE. The compressed package of Real-ESSI input files for this example is available HERE.

The Modeling parameters are listed below:

- **Soil**
  - Unit weight, $\gamma$, 21.4 kPa
  - Shear velocity, $V_s$, 500 m/s
  - Young’s modulus, $E$, 1.3 GPa
  - Poisson’s ratio, $\nu$, 0.25
  - Shear strength, $S_u$, 650 kPa
  - von Mises radius, $k$, 60 kPa
  - kinematic hardening, $H_a$, 30 MPa
  - kinematic hardening, $C_r$, 25

- **Structure**
  - Unit weight, $\gamma$, 24 kPa
  - Young’s modulus, $E$, 20 GPa
  - Poisson’s ratio, $\nu$, 0.21

- **Contact/Interface/Joint**
  - Initial axial stiffness, $k_n^{init}$, 1e9 N/m
  - Stiffening rate, $S_r$, 1000 /m
  - Maximum axial stiffness, $k_n^{max}$, 1e12 N/m
  - Shear stiffness, $k_t$, 1e7 N/m
  - Axial viscous damping, $C_n$, 100 $N \cdot s/m$
  - Shear viscous damping, $C_t$, 100 $N \cdot s/m$
  - Friction ratio, $\mu$, 0.25

**SIMULATION TIME**: With 32 cores on AWS EC2 c4.8xlarge instance, the running time for this example is 30 hours.
Figure 408.90: Simulation Model.
408.4.9 Buildings, ATC-144/FEMA-P-2091 Examples

The Real-ESSI building examples, models from FEMA-P-2091 report are available in Sections 509.2 (page 2554), 509.4 (page 2562), and 509.5 (page 2566), in Lecture Notes by Jeremić et al. (1989-2021) (Lecture Notes URL).
Chapter 409

Online Short Course Examples

(In collaboration with Dr. Han Yang, and Dr. Hexiang Wang)
409.1  Real-ESSI Simulator Setup
409.2 Phased Modeling of ESSI: Nonlinear Analysis Steps

Realistic Modeling of Earthquakes, Soils, Structures and their Interaction (Real-ESSI) requires careful, hierarchical modeling approach that is performed in phases. For a realistic 3D structure that is resting on soil, as seen in Figure 409.1.

Figure 409.1: Full 3D simulation model.
409.3  Week #1: Nonlinear/Inelastic Modeling Overview
409.4 Week #2: Discussion, Nonlinear/Inelastic Modeling
409.5  Week #3: Seismic Motions
409.6  Week #4: Discussion on Seismic Motions
409.7 Week #5: Inelastic Material and Contacts/Interfaces
409.8 Week #6: Discussion, Inelastic Material and Contacts/Interfaces
409.9 Week #7: Full Earthquake Soil Structure Interaction (ESSI) Analysis
Part 500

Application to Practical Engineering Problems
Chapter 501

Static Soil-Pile and Soil-Pile Group Interaction in Single Phase Soils

(In collaboration with Prof. Zhaohui Yang)
501.1 Chapter Summary and Highlights

501.2 Numerical Analysis of Pile Behavior under Lateral Loads in Layered Elastic–Plastic Soils

Material presented here has been previously published in our paper Yang and Jeremić (2005).

501.2.1 Introduction

The $p - y$ approach (Reese et al. Reese et al. (2000a)) has been widely used to design piles subjected to lateral loading. Based on the Winkler foundation theory, the method models the lateral soil-structure interaction with empirically derived nonlinear springs. The advancement of computer technology has made it possible to study this problem using more rigorous Finite Element Method (FEM).

Mentioned are a few representative finite element applications. Muqtadir and Desai Muqtadir and Desai (1986) studied the behavior of a pile-group using a three dimensional program with nonlinear elastic soil model. An axisymmetric model with elastic-perfectly plastic soil was used by Pressley and Poulos Pressley and Poulos (1986) to study group effects. Brown and Shie Brown and Shie (1990a) Brown and Shie (1990b) Brown and Shie (1991) and Trochanis Trochanis et al. (1991) conducted a series of 3D FEM studies on the behavior of single pile and pile group with elastic-plastic soil model. In particular, interface element was used to account for pile-soil separation and slippage. Moreover, Brown and Shie derived $p - y$ curves from FEM data, which provide some comparison of the FEM results with the empirical design procedures in use. A number of model tests of free- or fixed-headed pile groups under lateral loading has been simulated by Kimura et al. Kimura et al. (1995) and Wakai et al. Wakai et al. (1999) using 3D elasto-plastic FEM. A good correlation between the experiments and the analysis has been observed in these studies. All these results demonstrated that FEM can capture the essential aspects of the nonlinear problem. It is noted that there is not much literature reporting on FEM studies of pile behavior under lateral loading in layered soil system. In addition to that, there is a very small number of studies on the effects of layering system on the commonly used $p - y$ curve approach.

This paper describes four 3D finite element models of a laterally loaded pile embedded in uniform and layered soil profiles with the dimensions and soil parameters similar to those used in the centrifuge study by McVay et al. McVay et al. (1998) and Zhang et al. Zhang et al. (1999). The bending moments derived by integrating vertical stresses from FEM are numerically differentiated once and twice to compute the shear force and pressure diagrams, respectively. Particularly, $p - y$ curves are generated and cross compared to illustrate the effects of soft clay (sand) layer on the $p - y$ curves of the overlaid sand (soft clay) layer. The results from FEM are also compared with those from centrifuge test and
LPILE. In addition, a limited parametric study of pressure redistribution is conducted by changing the undrained shear strength of the soft clay layer and the friction angle of the sand layer to further investigate the layering effects. An early version of OpenSees OpenSees Development Team (Open Source Project) (2000-2006) finite element program was used in presented computations. Developed models are now available within our new framework F E I. Soil modeling was performed using Template Elastic–Plastic approach (Jeremić and Yang Jeremić and Yang (2002)).

501.2.2 Constitutive Models

Two simple models were used in this numerical study. Specifically, clay was modeled by a simple von Mises material model which is completely defined with the undrained shear strength. Sand was simulated by a Drucker–Prager material model with non-associated flow rule. The reason for using such simple models is that the experimental results used to compare our simulations against did specify only those two material properties for sands and clays. Figure 501.1 presents yield surfaces for both models. In both material models, the Young’s moduli vary with confining pressure, as shown in Eqn. (501.1).

\[
E = E_o \left( \frac{p}{p_a} \right)^\alpha
\]  

(501.1)

where \(E_o\) is Young’s Modulus at atmospheric pressure, \(p\) is the effective mean normal stresses, \(p_a\) is the atmospheric pressure, and \(\alpha\) is constant for a given void ratio. In this work, 0.5 was used.

![Diagram](a) Drucker–Prager model specified with friction angle and dilation angle, and (b) von Mises model specified with undrained shear strength \(C_u\).

The following parameters were used for medium dense sand: friction angle \(\phi\) of 37.1°, Shear modulus at a depth of 13.7 m of 8960 kPa \((E_o = 17400 \text{ kPa})\), Poisson’s ratio of 0.35 and unit weight of 14.50 kN/m\(^3\). These parameters were given by Zhang et al. Zhang et al. (1999). A dilation angle of 0° is used in this work (Brown and Shie Brown and Shie (1990a)). The undrained shear strength, Young's modulus, Poisson's ratio and unit weight of clay were chosen to be 21.7 kPa, 11000 kPa, 0.45, 13.7
\(kN/m^3\), respectively. It should be noted that the above material models are available within the Template Elastic–Plastic Material Modeling paradigm (Jeremić and Yang Jeremić and Yang (2002)). It should also be noted that the use of simple Drucker–Prager model can over-predict the friction angle to triaxial extension stress path. However this influence is limited to the zone behind the pile, within the interface zone and thus this drawback of the Drucker–Prager model was neglected.

### 501.2.3 Simulation Results

Presented in this subsection are representative results related to the behavior of piles in uniform and layered soil systems. Presented results are compared with those from the centrifuge study (McVay et al. McVay et al. (1998)), and with results obtained using LPILE program (Reese et al. Reese et al. (2000a,b)).

#### 501.2.3.1 Pile Models

A number of static pushover tests for single pile models were simulated using uniform soil and layered soil setups. Figure 501.2 shows the model setups. There are four main setups. Two of these are dealing with uniform sand and clay soils, while two others are featuring layered soils. In particular, the case #1

![Diagram](image_url)

Figure 501.2: (a) Single pile models, dimensions and layers of case #1 and #2. (b) Single pile models, dimensions and layers of case #3 and #4.
is a uniform soft clay soil, case # 2 includes top and bottom layers of soft clay with an in–between layer of medium dense sand. On the other hand, case # 3 features uniform medium dense sand soil, while case # 4 features top and bottom layers of medium dense sand with an in–between layer of soft clay. Detailed layering setup is given in Figure 501.2.

Figure 501.3 shows the finite element mesh for all four cases. Based on symmetry, only half of the model is meshed. Twenty node brick elements are used for both soil, pile and interface. It should be noted that these quadratic elements exhibit high accuracy even for high aspect ratios and can model accurately bending of solid piles with two layers of elements. During mesh design stage, a study was performed to decide on appropriate (balanced) mesh size. That study showed that a much larger mesh, with many more elements (with lower aspect ratios) would account for a fairly small change in results, so it was decided that the current mesh is sufficient for our analysis.

The square pile, with a width of 0.429 m, consist of four elements (per cross subsection) with the elastic property of aluminum. The fine mesh in the upper part of the model is to provide data points for the computation of shear forces and $p - y$ curves of sufficient reliability as well as for the investigation of layering effects. The sides and bottom of the model are fixed with the exception of the symmetric boundary, which is only supported in Y direction. The interface layer between aluminum pile and surrounding soil is represented by one thin layer of elements. The purpose of this layer is to mimic the installation effects on piles (drilled or driven). It also serves a purpose of a simplified interface which allows for tension cut-off (gaping) and controlled, coupled horizontal and vertical stiffness. All interface elements were simulated by Drucker–Prager model with a friction angle of $25^\circ$, and a dilation angle of $0^\circ$.

### 501.2.3.2 Plastic Zones

The static pushover test were conducted using load control at pile head. The final plastic zones are depicted in Figures 501.4, 501.5. Plastic zones are actually presented by plastified Gauss points. In particular, Figure 501.4(a) shows developed plastic zones for the uniform clay soil (case # 1). It is interesting to note that the plastic zone propagates fairly deep while it does not extend far from the pile in clay. Moreover, compression side (right side) features much larger plastic zone while the plastic zone for the extension side (left side) is confined to the interface layer and a few Gauss points outside the interface layer. The case with clay and sand layer in–between is shown in Figure 501.4(b). The main difference is that the plastic zone is even smaller than for uniform clay layer. It is worth mentioning that this case, which includes sand layer, is stiffer than the uniform clay case, thus displacements are smaller in clay and the plastic zone does not propagate as much as in uniform clay soil.

Figure 501.5(a)(b) shows plastic zones at the end of loading process for sand and sand and clay soils.
Figure 501.3: Mesh of single pile model, side view, top eight layers of finite elements are either clay or sand (depending on the cases), middle eight layers of finite elements are sand or clay (again depending on the cases) and the bottom is all uniform clay or sand, interface zone around the aluminum pile is also present.

In particular, Figure 501.5(a) shows the plastic zone for uniform sand. It is interesting to note that the plastic zone propagates toward the surface with the collapse mechanics similar to the active and passive failure. In this case of course the system is 3D and so the failure propagation angles do not match the active and passive failure angles, however the difference between active and passive zones propagation angles is almost exactly $\pi/2$. Figure 501.5(b) shows plastic zone for the case # 4 which includes a layer of clay between $-1.72$ m and $-3.44$ m (Z coordinate, origin is in the pile center at the ground surface). It is noted that the plastic zone is deeper, but not as nicely defined as in the previous case.

501.2.3.3 $p - y$ Curves

Results from static pushover tests on piles were used to generate $p - y$ curves. The bending moments derived by integrating vertical stresses are numerically differentiated once and twice to compute the shear force and pressure diagrams, respectively. Direct integration of shear stresses was also performed to check results and it was found that shear forces were within 5% accuracy. The combination of calculated
Figure 501.4: The plastic zones for (a) case # 1, and (b) case # 2 at lateral loading of 400kN.

Figure 501.5: The plastic zones of case 3 and 4 at lateral loading of 400kN.
pressures ($p$) and displacements obtained from the finite element solution, allowed for generation of $p-y$ curves at various depths along the pile.

In what follows, presented are generated $p-y$ curves for both uniform soils (sand and clay) as well as for layered systems. It is noted that the graphical presentation of results for bending moments, shear forces and lateral pressures (load) on a pile beam are shown with 10 lines, each one representing results for one increment ($1/10$) of the total load.

**Uniform Clay Soil.** Figure 501.6 shows bending moments, shear forces and pressures along the depth of a pile in clay soil. It should be noted that the maximum bending moment, as well as the switching of sign for shear force, moves quite a bit from the depth of approximately $-1.7m$ all the way to the depth of $-3.4m$. Pressure distribution shows that the top layers are already at the ultimate values of pressures and thus the pressure diagram propagates downward. There is a slight fluctuation of pressures at the depths of $4-5m$, which is attributed to the small numerical problems while doing double differentiations.

![Figure 501.6: Bending moment, shear force and pressure distributions for the uniform clay profile.](image)

Figure 501.6 shows generated $p-y$ curves for uniform clay layer. It is obvious that most of the clay (at least until the depth of $-2.6m$) has reached its peak resistance.
Uniform Sand Soil. Figure 501.8 shows bending moments, shear forces and pressures for a pile in a uniform sand soil. In this case it is interesting to note that the maximum bending moment, as well as the change of sign for the shear force is moving only between the depths $-1.8 \text{m}$ and $-2.0 \text{m}$. Moreover, the pressure diagram shows steady increase (with top layers reaching ultimate pressures) until the depth of $-1.7 \text{m}$ and then steadily decreases, and changes sign at greater depths (below $-4.0 \text{m}$).

Figure 501.9 shows generated $p-y$ curves for the uniform sand case. It is interesting to note that only the top layer at the depth of about $-0.3 \text{m}$ will reach the ultimate pressure. All the other sand material is far away from corresponding ultimate pressures. It is also worth noting that the displacements in the case of uniform sand are much smaller (almost twice as small) than what has been observed in uniform clay case.

Clay Soil with a Layer of Sand. Figure 501.10 shows bending moments, shear forces and pressures for a layered soil case. In this case a layer of sand extends from $-1.72 \text{m}$ to $-3.44 \text{m}$. The rest of soil is soft clay. It is interesting to note a large jump in pressures for the sand layer (as expected) and that the pressures in the top clay layer (from the surface to $-1.7 \text{m}$) reaches ultimate values. Small non–uniform distribution of the pressures at the interface of sand and clay at $-3.44 \text{m}$ is attributed to the coarseness of the finite element mesh. In comparing Figure 501.10 with the results for uniform clay case (Figure 501.6) it is obvious that the sand layer arrests the propagation of deformation and forces in depth and
Figure 501.8: Bending moment, shear force and pressure distributions for the uniform sand profile.

Figure 501.9: Calculated $p - y$ curves for the uniform sand profile.
Figure 501.10: Bending moment, shear force and pressure distributions for the clay soil with a sand layer.

fixes the maximum moment to approx. $-2.1m$.

Figure 501.11 shows generated $p - y$ curves for the layered case (single layer of sand in clay). The $p - y$ curves were generated only for the top layer of clay and middle layer of sand, to the depth of $-2.7m$. It is interesting to note that the $p - y$ curve for clay at the depth of $-1.61m$ (close to the sand layer) exhibits strong hardening, unlike similar curve for the uniform clay soil, in Figure 501.7. The increase in pressure (transversal loading on the pile) between uniform clay (Fig. 501.7) and clay underlain by a medium dense sand layer (Fig. 501.11) at the displacement of 0.06m is more than two times.

Sand Soil with a Layer of Clay. Figure 501.12 shows bending moments, shear forces and transversal pressures for a case where a layer of soft clay is present within sand soil. Unlike the case of uniform sand soil (Figure 501.8) the presence of soft clay layer will change the depth of maximum moment by almost 1m (from $-2.0m$ to $-3.0m$). In addition to that, the distribution of pressures on a pile is changed significantly, as seen in the right plot of Figure 501.12. The reduction of pressures will extend into the sand layer and present significant influence of soft clay on pressures in sand.

Figure 501.13 shows generated $p - y$ curves for the case of sand with a soft clay layer. It is noted that the $p - y$ curves for sand that is some distance away from the interface with clay are much the same as for the uniform sand case (refer to Fig. 501.9 and Fig. 501.19(a)). However, the $p - y$ curves in sand
Figure 501.11: Calculated $p - y$ curves for the clay soil underlain by a medium dense sand layer.

Figure 501.12: Bending moment, shear force and pressure distributions for the sand soil with a soft clay layer.
close to the interface are changed in some cases significantly. For example, the \( p - y \) curve at depth of \(-1.61\,\text{m}\) is showing pressure of approx. \( p = 265\,\text{kN/m} \) at the displacement of \(0.042\,\text{m} \) for the uniform sand case, while the same \( p - y \) curve, still in sand, has a drop in pressure at the same displacement to \( p = 140\,\text{kN/m} \). Similar trend is observed for other \( p - y \) curves close to the interface of sand with clay.

![Figure 501.13](image)

Figure 501.13: Calculated \( p - y \) curves for the sand soil underlain by a soft clay layer.

### 501.2.3.4 Comparisons of Pile Behavior in Uniform and Layered Soils

Comparison of pile behavior in uniform and layered soils can also be performed by looking at the displacement and bending moment distributions. For example, Figure 501.14 compares the distributions of displacements for the uniform sand case with the sand and clay layer case. First observation is that the uniform sand layer allows smaller displacements of the pile head (\(0.12\,\text{m}\)) while the inclusion of clay layer raises those displacements to \(0.22\,\text{m}\). Second observation is that the point of rotation for the pile (point which does not move as the loading is applied) is pushed deeper, from \(5\,\text{m}\) to approximately \(6\,\text{m}\). Moreover, the propagation of displacements along the depth of a pile is much greater for a layered case, the surface displacement is extended from \(0.09\,\text{m}\) to almost \(0.13\,\text{m}\).

Figure 501.15 shows similar results for uniform clay and clay with a layer of sand case. In this case, the inclusion of a sand layer will increase the stiffness of the pile (as expected) and will also reduce propagation of displacements with depth.
Figure 501.14: Pile displacement distributions along the depth in a uniform sand profile (left) and sand with clay layer profile (right).

Figure 501.15: Pile displacement distributions along the depth in a uniform clay profile (left) and clay with sand layer profile (right).
Figure 501.16 shows comparison of pile head displacements for all four cases. It is noted that the two layered cases exhibit similar behavior in terms of displacements, both at the pile head and in terms of displacement profiles (compare right plot in Fig. 501.14 and left plot in Fig. 501.15).

![Figure 501.16: Pile head displacement comparison for all the four cases.](image)

Figure 501.17 shows comparison of the maximum bending moment calculated for the pile for all four cases. It is interesting to note that the difference between the two uniform soil cases (uniform sand and uniform clay) is not that pronounced. Of course one has to remember that the material for pile was assumed to be linear elastic, no yielding was allowed for the aluminum pile.

The $p-y$ curves for uniform clay and clay with a layer of sand were plotted together in Figure 501.18 (a) for comparison. It can be seen that all the $p-y$ curves in clay except the one right next to the layer interface are almost identical. In order to measure the magnitude of the effects of sand layer on the pressure of soft clay layer, the ratio of pressures in clay layer for clay soils with a sand layer and uniform clay soils lateral displacement of 12\%D, i.e. 5.15 cm, were computed and plotted against the distance in terms of times of pile width D in Figure 501.18. It is noted that the disturbance to the pressure field is much more confined to the immediate vicinity (within 0.75D) of the layer interface. In addition, the results from two more analysis of the same model with different sands (friction angles $\phi' = 25^\circ, 30^\circ$ respectively, other parameters remain the same.) were included in Figure 501.18. It is shown that the lateral pressure ratio is affected considerably when sand friction angle increases from $25^\circ$ to $37^\circ$ (from 1.5 times to 2.2 times more pressure).

The $p-y$ curves for uniform sand and sand with a layer of soft clay were also plotted together for
comparison purposes. It was found that the effect of soft clay on the pressures in sand propagates far away from the layer interface. Therefore, three cases of an additional model with a thicker sand layer (2.4m in thickness) underlain by a soft clay layer were analyzed by varying the undrained shear strength ($C_u = 13.0 \text{ kPa}, \ 21.7 \text{ kPa}, \ 30.3 \text{ kPa}$) of the soft clay layer. Similarly, the pressure ratios at 6.5% D, i.e. 2.8 cm, were plotted in Figure 501.19. It is noted that the effects extends to as far as 4.75D from the layer interface and the reduction of pressures adjacent to the interface is about 0.6 in all three cases.
Figure 501.18: (a) Comparison of $p - y$ curves for uniform clay versus clay with a layer of sand ($\phi' = 37^\circ$).
(b) Pressure ratio distributions in clay layer for sands with different friction angle ($\phi' = 25^\circ, 30^\circ, 37^\circ$).
Figure 501.19: (a) Comparison of $p - y$ curves for uniform sand versus sand with a layer of soft clay ($C_u = 21.7 \text{ kPa}$). (b) Pressure ratio distributions in sand layer for clays with different undrained shear strength ($C_u = 13.0 \text{ kPa}, \ 21.7 \text{ kPa}, \ 30.3 \text{ kPa}$).
501.2.3.5 Comparison to Centrifuge Tests and LPILE Results

The pile head displacements for uniform sand profile from 3D FEM, LPILE (Reese et al. Reese et al. (2000a,b)), and centrifuge test (McVay et al. McVay et al. (1998)) were plotted against pile head load in Figure 501.20. It can be seen that they agree with each other fairly well. It should be noted that the material properties for our 3D finite element simulations were not in any particular way calibrated to improve the results. They were simply used as presented in the centrifuge study by McVay et al. McVay et al. (1998) and numerical simulation by Zhang et al. Zhang et al. (1999). Whereas, the results from LPILE were back-fitted since the coefficient of subgrade reaction $\eta_h$ was back-calculated as 2714 kN/m$^3$ (Zhang et al. Zhang et al. (1999)).

![Figure 501.20: Simulated versus experimental pile head displacements.](image)

The bending moments, shear forces and lateral pressures of uniform sand and clay profiles from 3D FEM and LPILE were plotted against pile depth at several pile head loads in Figure 501.21 and 501.22. In general, there is a good agreement between the results from FEM and LPILE in uniform sand profile. In uniform soft clay profile, it is noted that the pressures at shallow depth from LPILE are smaller than those computed by FEM, which agrees with one of the findings by the work of Steven and Audibert Stevens and Audibert (1979). For example, the pressures at lateral load of 120kN and 200kN from LPILE are only about half of those from FEM. Because the pressures at shallow depths are so small in LPILE that the pile head has to deform much more than in FEM and the passive pressure zone in LPILE
extends to fairly large depth.

Since LPILE currently uses the equivalent depth method developed by Geogiadis (1983) for layered soil profiles, the LPILE output pressure distribution along pile depth, especially across the layer interface does not take into account of the layering effect, thus it is not that meaningful to compare pressure distributions of layered profiles from LPILE versus FEM.

![Graph](image)

Figure 501.21: Comparison of bending moment, shear force and pressure computed by FEM and LPILE in uniform sand profile (case #3).

It is also interesting to compare the $p-y$ curves derived from FEM with those used in LPILE. Figures 501.24 and 501.23 show FEM derived and LPILE used $p-y$ curves for uniform clay and sand profiles, respectively. It should be noted that the coefficient of subgrade reaction $\eta_h$ was again back-calculated as 8969 kN/m$^3$ in order to get a reasonable $p-y$ curves. From Figures 501.24 (a) and (b), it is clear that $p-y$ curves in sand profile from LPILE have lower resistance at depth close to ground surface. The $p-y$ curves for clay profile shown in Figures 501.24 (a) and (b) are seen to have much lower resistance at shallow depths.

501.2.4 Summary

This paper presents results from a finite element study on the behavior of a single pile in elastic–plastic soils. The analysis included single pile behavior in sand, clay and layered soils. Based on the results
presented, it is concluded that three dimensional finite element analysis using very simple elastic-plastic soil models can predict the pile head deflection with very good accuracy.

The main findings of this numerical study can be summarized as follows:

- When a sand layer is present within a clay deposit, the increase in lateral pressure in clay near the interface is confined to a narrow zone, up to two times of pile width, therefore the layering effect in this case is not prominent.

- When a clay layer is present within a sand deposit, the reduction in pressures spread well into the sand layer (up to four times of pile width). The layering effects are of more importance in this case since the disturbance zone is large and the pressure reduction is significant. Reduction factors are given in terms of charts of pressure reduction versus the distance from the interface.

In addition, comparison with centrifuge data shows generally a good agreement between the bending moments, shear forces and lateral resistance. Moreover, a comparison with results from program LPILE, used in extensively in practice, show some discrepancies ultimate pressures in shallow soil layers.

Figure 501.22: Comparison of bending moment, shear force and pressure computed by FEM and LPILE in uniform soft clay profile (case #1).
Figure 501.23: $p - y$ curves from FEM (a) and LPILE (b) in uniform sand profile ($\eta_h = 8969 \text{ kN/m}^3$, $\phi = 37.1^\circ$).
Figure 501.24: $p - y$ curves from FEM (a) and LPILE (b) in uniform clay profile ($\varepsilon_{50} = 0.02$, $C_u = 21.6 \text{kPa}$).
501.3 Study of soil layering effects on lateral loading behavior of piles

Material presented here has been previously published in our paper Yang and Jeremić (2005).

501.3.1 Introduction

The theory of beams on a Winkler-type subgrade (Hartog (1952)), also known as the $p$--$y$ approach, has been widely used to design piles subjected to lateral loading. Based on that theory, the method models the lateral soil–foundation interaction with empirically derived nonlinear springs ($p$--$y$ curves). The advancement of computer technology has made it possible to study this problem using more rigorous elastic–plastic Finite Element Method (FEM).

Here mentioned are a few representative examples of finite element studies of pile foundations. Muqtadir and Desai (1986) studied the behavior of a pile–group using a three dimensional (3D) program with nonlinear elastic soil model. An axisymmetric model with elastic-perfectly plastic soil was used by Pressley and Poulos (1986) to study group effects. Brown and Shie (1990a), Brown and Shie (1990b), Brown and Shie (1991), and Trochanis et al. (1991) conducted a series of 3D FEM studies on the behavior of a single pile and a pile group with elastic-plastic soil model. These researchers used interface elements to account for pile–soil separation and slippage. Moreover, Brown and Shie derived $p$--$y$ curves from FEM data, which provide some comparison of the FEM results with the empirical design procedures in use. Kimura et al. (1995) conducted 3D FEM analysis of the ultimate behavior of laterally loaded pile groups in layered soil profiles with the soil modeled by Drucker–Prager model and pile modeled by nonlinear beam elements. A number of model tests of free– or fixed–headed pile groups under lateral loading in homogeneous soil profiles have been simulated by Wakai et al. (1999) using 3D elasto-plastic FEM. Pan et al. (2002) studied the performance of single piles embedded in soft clay under lateral soil movements. A good correlation between the experiments and the analysis has been observed in these studies. All these results demonstrated that FEM can capture the essential aspects of the nonlinear problem.

Information about the lateral behavior of piles in layered soil profiles is very limited. Some analytical studies have been conducted by Davisson and Gill (1963) and Lee and Karunaratne (1987) to define the influence of pile length, the thickness of upper layer and the ratio of stiffness ratio of adjacent layers on the pile response based on the assumption that the soil is elastic. Reese et al. (1981) conducted small scale laboratory tests on a 25 mm diameter pile and a field test with 152 mm diameter pile in layered soils and found that there was a relatively good agreement between deflections measured in the tests and deflections computed using homogeneous $p$--$y$ curves at small loads. Georgiadis (1983) proposed an approach which is currently used in the LPILE program (Reese et al. (2000a,b)). This method assumes
the $p-y$ curves of the first layer are the same as those for homogeneous soils. The effects of upper layers on the $p-y$ curves of the lower layers are accounted for by the equivalent depth of the overlying layers based on strength parameters.

To the Authors’ knowledge, there is no literature reporting on FEM study of layering effects on the behavior of laterally loaded piles in layered profiles. However, it is of great interest to investigate the layering effects since in practice, most of soil deposits are layered systems. In a predominantly clay site with a minor sand layer, the sand layer will still be counted on to provide most of the soil resistance. In this case, the layering effects (probably reduction of resistance in the sand layer) must be considered. Current practice is to “make an educated guess to reduce the sand $p-y$ curves to account for the soil layering effects” (Lam and Law (1996)). Obviously, an educated guess might not result in optimal design. It is very important to find out how these layers in the layered system affect each other in order to carry out a more accurate analysis of pile foundation and therefore provide a more effective way for the design of pile foundations in layered soil systems.

This paper describes four 3D finite element models of a laterally loaded pile embedded in uniform and layered soil profiles, with the dimensions and soil parameters similar to those used in the centrifuge studies by McVay et al. (1998) and Zhang et al. (1999). Visualization tool Joey3D (Yang (2002)) was used to compute the bending moment, shear force and lateral resistance diagrams along the pile. Model calibration, comparison of finite-element analysis results with those from centrifuge tests and the LPILE program, and comparison of finite-element generated $p-y$ curves with traditional $p-y$ curves are summarized in a separate paper (Yang and Jeremić (2003)). In this paper, $p-y$ curves from each model were cross compared to illustrate both the effects of an intermediate soft clay (or sand) layer on the $p-y$ curves of the sand (or soft clay) layers and the effects of sand (or soft clay) layers on the intermediate soft clay (or sand) layer. In addition, a limited parametric study was conducted to further investigate the layering effects in terms of lateral resistance ratios. The OpenSees OpenSees Development Team (Open Source Project) (2000-2006) finite element framework was employed for all the computations. Soil modeling was performed using the Template Elasto–Plastic Framework (Jeremić and Yang (2002)) and solid elements while the piles were modeled using linear elastic solid elements, all developed by the Authors.

501.3.2 Finite Element Pile Models

Single pile finite element models with the dimensions similar to the prototype model described in the above centrifuge tests were developed and a number of static pushover tests were simulated with 3D FEM using uniform soil and layered soil cases. The models for all cases were illustrated in Figure 501.25 (a). There are four main analysis models. Two of them are dealing with uniform sand and clay deposits,
Figure 501.25: (a) Single pile models, dimensions and layers for models #1, #2, #3 and #4, including pile-soil interfaces and (b) 3D mesh of the single pile model.

while the other two are featuring layered soil deposits. In particular, model # 1 has a uniform soft clay deposit, model # 2 includes top and bottom layers of soft clay with an interlayer of medium dense sand. Model # 3 features uniform medium dense sand deposit, while model # 4 features top and bottom layers of medium dense sand with an interlayer of soft clay.

Figure 501.25 (b) shows the finite element mesh for all four models. Based on symmetry, only half of the model is meshed. Twenty–node brick elements are used to mesh the soil, pile and pile–soil interface. The square pile, with a width of 0.429 m and length of 13.7 m, is divided into four elastic elements (per cross subsection) with the properties of aluminum. The mesh is refined at the upper part of the model in order to provide data points for the computation of shear force and lateral resistance of sufficient reliability as well as for investigation of the layering effects. Additional finite element analysis of a cantilever beam using the same mesh as the pile was carried out and comparison of the beam displacement from FEM and beam theory solution indicated that the mesh was fine enough to capture the pile behavior. As to the boundaries, the sides and bottom of the model are fixed with the exception of the symmetric boundary, which is only supported in Y direction. Since the sides are 13 times of the pile width away from the pile center, it is believed that the fixed boundaries have very limited effects on the results. In addition to that the model size is closely following that of the physical, centrifuge model, which resided in a container of similar size. The pile–soil interface is represented by one thin layer of elements. The purpose of this layer is to mimic the installation effects on the pile (drilled or driven). It

\[ \text{Jeremić et al.} \quad \text{University of California, Davis} \quad \text{version: 28. May, 2021, 17:09} \]
also serves a purpose of a simplified interface which allows for tension cut-off (gapping) and controlled, coupling of horizontal and vertical resistance according to Coulomb frictional laws.

501.3.3 Constitutive Models

Two simple models were used in this numerical study. Specifically, clay was modeled by von Mises material model which is completely defined with the undrained shear strength. Sand was simulated by Drucker–Prager material model with nonassociated flow rule, defined with the friction and dilation angles. The reason for using such simple models is that the experimental results used in comparison with simulations did specify only very limited number of material properties for sands. Furthermore, a small number of model parameters needed by simple models are convenient for parametric study. In both material models, the Young’s moduli vary with confining pressure, as shown in Eqn. (501.2) (cf. Janbu (1963), Duncan and Chang (1970)):

\[ E = E_o \left( \frac{p}{p_a} \right)^a \]

(501.2)

where \( E_o \) is Young’s Modulus at atmospheric pressure, \( p \) is the effective mean normal stresses, \( p_a \) is the atmospheric pressure, and \( a \) is constant for a given void ratio. In this work, 0.5 was used.

The following parameters were used for medium dense sand: friction angle \( \phi = 37.1^\circ \), Shear modulus \( G \) at a depth of 13.7 m = 8960 kPa \( (E_o = 17400 \) kPa), Poisson’s ratio \( \nu = 0.35 \) and unit weight \( \gamma = 14.50 \) kN/m\(^3\). These parameters were given by Zhang et al. (1999). A dilation angle of \( \psi = 0^\circ \) is used in this work (Brown and Shie (1990a)). The undrained shear strength, Young’s modulus, Poisson’s ratio and unit weight of clay were chosen to be \( C_u = 21.7 \) kPa, \( E_0 = 11000 \) kPa, \( \nu = 0.45 \), \( \gamma = 11.8 \) kN/m\(^3\), respectively. The interface elements were simulated by Drucker–Prager model with a friction angle \( \phi = 25^\circ \), and a dilation angle \( \psi = 0^\circ \). All material properties were summarized in Table 501.1.

Table 501.1: Material properties of sand, clay, pile and soil–pile interface used in FEM analysis.

<table>
<thead>
<tr>
<th>Soil</th>
<th>( E_o ) (kPa)</th>
<th>( \nu )</th>
<th>( \gamma ) (kN/m(^3))</th>
<th>( \phi ) (°)</th>
<th>( \psi ) (°)</th>
<th>( C_u ) (kPa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium dense sand</td>
<td>17400</td>
<td>0.35</td>
<td>14.5</td>
<td>37.1</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>Loose sand</td>
<td>16000</td>
<td>0.35</td>
<td>14.1</td>
<td>34.5</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>Clay</td>
<td>11000</td>
<td>0.47</td>
<td>11.8</td>
<td>–</td>
<td>–</td>
<td>21.7</td>
</tr>
<tr>
<td>Pile</td>
<td>69000000</td>
<td>0.33</td>
<td>26.8</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Soil–pile interface</td>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
<td>25</td>
<td>0</td>
<td>–</td>
</tr>
</tbody>
</table>
501.3.4 Comparison of $p$–$y$ Behavior in Uniform and Layered Soil Deposits

This subsection presents representative results related to the behavior of piles in uniform and layered soil deposits. Specifically the $p$–$y$ response curves derived for 3D FEM results for homogeneous and layered soil deposits are compared with each other to investigate the layering effects.

501.3.4.1 Uniform Clay Deposit and Clay Deposit with an Interlayer of Sand.

The $p$–$y$ curves of uniform clay deposit and clay deposit with a layer of sand were compared in Figure 501.26. It is clearly seen that the $p$–$y$ curve ($Z = -3.75D$) close to the interface ($Z = -4D$) is significantly different from that in uniform soil profile.

![Comparison of $p$–$y$ curves](image)

Figure 501.26: Comparison of $p$–$y$ curves of uniform clay deposit versus clay deposit with an interlayer of sand (Sand: $\phi = 37^\circ$; Clay: $C_u = 21.7$ kPa).

In order to measure the magnitude of the effects of the intermediate sand layer on the lateral resistance of the soft clay layers and vice versa, the ratios of soil lateral resistances in the layered ($p$) and uniform models ($p_{\text{homog. model}}$) at several lateral displacements (i.e. 0.5%, 1.0%, 2.0%, 2.5%, 8.0% and 10.0% of pile width $D$) were computed and plotted against vertical coordinate ($Z$) normalized by...
pile width $D$ in Figures 501.27 and 501.28. In addition, the results from two more analyses of the same model with different sands (friction angle $\phi$ were varied from $25^\circ$ to $30^\circ$, while originally, the friction angle was set to $37^\circ$) were also included in these figures.

From Figure 501.27, it is observed that the lateral resistance ratios are independent of friction angle $\phi$ of sand at small lateral displacements ranging from 0.5% to 1.0% of pile width $D$. When the lateral displacement is greater than 1.0%, the variation in $\phi$ starts to affect the lateral resistance ratio, as shown in Figure 501.28.

![Figure 501.27: Lateral resistance ratio distributions (Clay:$C_u = 21.7$ kPa, $E_o = 11000$ kPa) for sands with various $\phi$ at lateral displacements of 0.5% and 1.0% pile width.](image)

Overall, the effect of the sand layer reduces to less than 10% at about one pile width above the upper sand interface and the lateral resistance ratio at a quarter pile width above the upper sand interface is 1.3 at lateral deflection of 0.5% pile width. It may be noted that the two dashed vertical lines in the lateral resistance plots correspond to lateral resistance ratios of 0.9 and 1.1, indicating $\pm 10\%$ change in lateral resistance. The 10% change will be used to judge the extent of influence throughout the rest of the paper. The resistance ratio below the lower sand interface was not processed since the mesh is becoming coarse and the results are affected by mesh effects and numerical differentiation, and the pile displacements are very small.

Besides the effect the sand interlayer has on the clay layer, it is interesting to observe in Figure 501.27

---

2The lateral resistance ratio is only shown for the upper clay layer since the resistance corresponding to large $y$ is not available at larger depth due to the fact that the pile is loaded at the pile head and the deflection decreases quickly as depth increases. Also due to the limit of space, plots for 2.0%, 2.5% pile width are not shown in this paper.
that the soft clay layers also have significant effect on the lateral resistance of the intermediate sand layer. The lateral resistance ratios are less than 0.9 throughout the interlayer of sand. Surprisingly, the effects are not symmetric at lateral deflection of 0.5%. The resistance ratio is 0.85 at 0.25D below the upper sand interface, while that is 0.72 at 0.25D above the lower sand interface. This non–symmetry is probably due to the non–symmetric deformation\(^3\) mode in the pile. As the pile is loaded laterally at the pile head, the right–hand–side sand close to the pile below certain depth tends to move downward to the right, which can be observed in Figure 501.32 (b). Therefore, the sand close to the upper interface moves against sand, while that close to the lower interface moves against soft clay. This type of movement results in the larger reduction in resistance at the lower sand interface than at the upper sand interface. The decrease in lateral resistance is mainly due to the lower stiffness in the adjacent soft clay layers. In addition, the smaller unit weight of the soft clay results in smaller mean effective normal stresses in the sand layer than the homogeneous model, which will reduce the stiffness of the sand and therefore also contribute to the reduction in lateral resistance at the intermediate sand layer.

\(^3\)Non–symmetric with respect to the horizontal plane in between the interfaces (midway through the sand layer).
501.3.4.2 Uniform Sand Deposit and Sand Deposit with an Interlayer of Soft Clay.

By comparing the $p-y$ curves of uniform sand deposit and sand deposit with an interlayer of soft clay, it was found that the effect of soft clay on the lateral resistance of sand propagates further away from the interface than Clay-Sand-Clay case, as described above in subsection 501.3.4.1. In addition to that, it was found that the heave in front of the pile will affect the lateral resistance of sand at shallow depth. Therefore, for sand deposit with an interlayer of soft clay, the thickness of upper sand layer was increased from 1.72 m to 2.36 m (the thickness of the soft clay layer was kept the same) to investigate the range of layering effects. Three models were analyzed by only varying the undrained shear strength $C_u$ (i.e. 13.0, 21.7 and 30.3 kPa) of the soft clay layer.

Similar to the previous analysis, the $p-y$ curves from the uniform deposit and the re-configured layered deposit were compared in Figure 501.29 and the lateral resistance ratios at several lateral displacements (i.e. 0.5%, 1.0%, 2.0%, 2.5%, 5.0% and 6.5% of pile width $D$) for all three models were computed and shown in Figures 501.30 and 501.31. It may be observed from Figure 501.29 that obvious difference may be observed in several $p-y$ curves further away from the interface.

Figure 501.29: Comparison of $p-y$ curves for uniform sand deposit versus sand deposit with an interlayer of soft clay (Sand: $\phi = 37^\circ$; Clay: $C_u = 21.7$ kPa).
Figure 501.30: Lateral resistance ratio distributions (Sand: $\phi = 37^\circ$, $E_o = 17400$ kPa) for clays with various $C_u$ at lateral displacements of 0.5% and 1.0% pile width.

Figure 501.31: Lateral resistance ratio distributions (Sand: $\phi = 37^\circ$, $E_o = 17400$ kPa) for clays with various $C_u$ at lateral displacements of 5.0% and 6.5% pile width.
From Figure 501.30, it is noted that the effects of the intermediate soft clay layer are also independent of its undrained shear strength at small lateral displacements ranging from 0.5% D to 1.0% D. When the lateral displacement is greater than 1.0% D, the change in $C_u$ starts to affect the lateral resistance ratio, as shown in Figure 501.31. Similar to the Clay–Sand–Clay model, the effect of the intermediate soft clay layer reduces to less than 10% at one pile width above the clay interface. The lateral resistance ratio at 0.25D above the clay interface is about 0.75. For large lateral displacements ranging from 5.0% D to 6.5% D, the 10% change in lateral resistance extends to 1.5 D - 2 D, as can be observed in Fig. 501.31. It may be noted that, at a lateral displacement of 6.5% D, the lateral resistance ratio at 0.25D above the clay interface changes from 0.58 to 0.67 when $C_u$ increases from 13.0 kPa to 30.3 kPa.

Figures 501.32 (a) and (b) show the details of displaced models around the interfaces for the Sand–Clay–Sand and Clay–Sand–Clay profiles, respectively. The deformed model was overlapped with undeformed model for comparison. Ground heave can be easily observed in front of the pile from both figures. It is noted from Figure 501.32 (a) that the sand crosses the upper clay interface and moves into the intermediate soft clay layer. The movement slightly strengthens the soft clay soil and partially causes the slight increase of lateral resistance at the top of soft clay layer. Most importantly, the movement will soften the sand close to the upper layer interface, due to the reduction of confinement to the sand. For the Clay–Sand–Clay profile, the stronger sand layer penetrates into the softer clay layers at both interfaces. This penetration softens the sand close to both interface, due to the same reason as above.

501.3.5 Parametric Study for the Lateral Resistance Ratios in Terms of Stiffness and Strength Parameters.

To further investigate the effects of soil stiffness on the lateral resistance ratios at small displacement and/or large displacement, further analyses were carried out for the Clay–Sand–Clay and Sand–Clay–Sand models by changing both stiffness parameter (i.e. $E_o$) and strength parameter ($C_u$ for clay, or $\phi$ for sand) using the same finite element models as above. The model configurations and intermediate layer soil parameters were summarized in Tables 501.2 and 501.3.

Lateral resistance ratios were plotted in Figures 501.33 and 501.34 for the Clay–Sand–Clay model, and in Figures 501.35 and 501.36 for the Sand–Clay–Sand model. By comparing Figures 501.28 and 501.34 for pile displacements of 8% D and 10% D, and Figures 501.31 and 501.36 for pile displacements of 5% D and 6.5% D, it is clear that the lateral resistance ratios are almost the same for the upper layer soil even if the stiffness parameter $E_o$ of intermediate layer soil was varied by more than 30%. However, the lateral resistance ratios at small displacement (0.5% D and 1.0% D) were obviously influenced by the variation of $E_o$, as can be observed by comparing Figures 501.27 and 501.33 for the Clay–Sand–Clay model, and Figures 501.30 and 501.35 for the Sand–Clay–Sand model. For medium pile displacements
Figure 501.32: Details of displaced model indicating ground heave and movement of soils across the layer interfaces at lateral load of 400 kN: (a) sand deposit with an interlayer of soft clay and (b) clay deposit with an interlayer of medium sand. The pile elements are removed so that the interface layer in the middle can be seen clearly.

Table 501.2: Summary of model configurations and intermediate sand layer parameters for Clay–Sand–Clay model in the parametric study.

<table>
<thead>
<tr>
<th>Case</th>
<th>Soil Profile</th>
<th>Depth of Interfaces</th>
<th>Intermediate Sand Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Upper</td>
<td>Lower</td>
</tr>
<tr>
<td>1</td>
<td>Clay–Sand–Clay</td>
<td>-1.72 m</td>
<td>-3.43 m</td>
</tr>
<tr>
<td>2</td>
<td>Clay–Sand–Clay</td>
<td>-1.72 m</td>
<td>-3.43 m</td>
</tr>
<tr>
<td>3</td>
<td>Clay–Sand–Clay</td>
<td>-1.72 m</td>
<td>-3.43 m</td>
</tr>
</tbody>
</table>
Table 501.3: Summary of model configurations and intermediate clay layer parameters for Sand–Clay–Sand model in the parametric study.

<table>
<thead>
<tr>
<th>Case</th>
<th>Soil Profile</th>
<th>Depth of Interfaces</th>
<th>Intermediate Clay Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Upper m</td>
<td>Lower m</td>
</tr>
<tr>
<td>1</td>
<td>Sand–Clay–Sand</td>
<td>-2.36 m</td>
<td>-4.08 m</td>
</tr>
<tr>
<td>2</td>
<td>Sand–Clay–Sand</td>
<td>-2.36 m</td>
<td>-4.08 m</td>
</tr>
<tr>
<td>3</td>
<td>Sand–Clay–Sand</td>
<td>-2.36 m</td>
<td>-4.08 m</td>
</tr>
</tbody>
</table>

(e.g. 2% D and 2.5% D), both stiffness and strength parameters have effects on the lateral resistance ratios.

Figure 501.33: Lateral resistance ratio distributions (Clay: $C_u = 21.7$ kPa, $E_o = 11000$ kPa) for sands with various $\phi$ and $E_o$ at lateral displacements of 0.5% and 1.0% pile width.

It will be useful to relate the effects of (a) the relative stiffness which controls the lateral resistance ratio at small lateral displacements and (b) the relative strength which determines the lateral resistance ratio at large lateral displacements with the lateral resistance ratio. To exclude the effects of unit weight, only the results above the upper interface in the Sand–Clay–Sand model are processed. The ratio of Young’s moduli of clay and sand soils was used to define the relative stiffness $R_{stiffness}$ of the two
Figure 501.34: Lateral resistance ratio distributions in clay layer ($C_u = 21.7$ kPa, $E_o = 11000$ kPa) for sands with various $\phi$ and $E_o$ at lateral displacements of 8% and 10% pile width.

Figure 501.35: Lateral resistance ratio distributions (Sand: $\phi = 37^\circ$, $E_o = 17400$ kPa) for intermediate layer of clays with various $C_u$ and $E_o$ at lateral displacements of 0.5% and 1.0% pile width.
layers. On the other hand, the ratio of largest lateral resistances of uniform clay and sand\(^4\) at the upper interface (-2.36 m) was used to define the relative strength \(R_{\text{strength} - \text{FEM}}\), as described in Equations (501.3) and (501.4).

\[
R_{\text{stiffness}} = \frac{E_{o - \text{clay}}}{E_{o - \text{sand}}}
\]  

(501.3)

\[
R_{\text{strength} - \text{FEM}} = \frac{p_{\text{clay} - \text{FEM}}}{p_{\text{sand} - \text{FEM}}}
\]  

(501.4)

The lateral resistance ratios at lateral displacement of 6.5% \(D\) were plotted against \(C_u\) in Figure 501.37. For comparison, the relative stiffness \(R_{\text{stiffness}}\) and relative strength \(R_{\text{strength} - \text{FEM}}\) were also included in the same plot.

As can be observed from this plot, the lateral resistance ratio decreases from 0.69 to 0.56 almost proportionally as \(C_u\) drops from 30 kPa to 13 kPa at 0.25 \(D\) above the upper interface, and the ratio is greater than the relative strength \(R_{\text{strength} - \text{FEM}}\). Since the ultimate resistance of uniform sand will be larger than the computed largest value (which is still increasing), as can be observed from Figure 501.29 at \(Z = -3.75D\) and that of uniform clay almost will almost remain the same (refer to Figure 501.36).

\(^4\)It would be better to use the ultimate lateral resistances for both clay and sand to define the relative strength but these values are not available from the current numerical results since pile displacement \(y\) is not large enough.
Figure 501.37: Lateral resistance ratios in the upper sand layer ($\phi = 37^\circ$) at various distances from the interface for pile displacement of 6.5% pile width.

501.26 at $Z=-3.75D$), this relative strength value will drop and the above statement still holds. There is certain correlation between the lateral resistance ratio close to the upper interface and $R_{\text{strength-FEM}}$ at 6.5%D pile displacement. As the distance to the upper interface increases, this correlation diminishes. The relative stiffness curve intercepts with the lateral resistance ratio curves at 0.25D above the upper interface. This implies that the presence of the clay, which is softer than the sand, somehow caused the layered system to be softer than either of the homogeneous models. This seems illogical, and in fact previous discussions and comparisons showed that $R_{\text{strength-FEM}}$ is more important than $R_{\text{stiffness}}$ at these large relative displacements.

It is also interesting to examine the relationship between the lateral resistance ratio and the relative variables (i.e. strength and stiffness) when lateral displacement increases, as presented in Figures 501.38 and 501.39. Figure 501.38 shows that the lateral resistance ratios at 0.25D above the interface decreases and come closer to the relative strength $R_{\text{strength-FEM}}$ curve as the lateral displacement increases from 4.0%D to 6.5%D. The relative stiffness $R_{\text{stiffness}}$ was also plotted in Figure 501.38 and it intercepts with the lateral resistance ratio curve, which has similar implications as the above discussion for Figure 501.37 and is illogical. On the other hand, as the lateral displacement decreases from 1.5%D to 0.5% D, the lateral resistance ratios keep decreasing and come closer to the relative stiffness ratio $R_{\text{stiffness}}$, as shown in Figure 501.39. There is almost a linear relationship between the lateral resistance ratio and the relative stiffness at small displacements.
Figure 501.38: Lateral resistance ratio at a quarter pile width above the upper clay interface for various deflections.

Figure 501.39: Lateral resistance ratio at one quarter pile width above the upper clay interface for clays with various $C_u$ and $E_o$ at small pile displacements ranging from 0.5% to 1.5% pile width.
From the above analysis, it is safe to say that the lateral resistance ratio is dominated by the relative stiffness $R_{stiffness}$ at small displacement (i.e. $\leq 0.5\%D$), while that is controlled by the relative strength $R_{strength-FEM}$ at large displacement (i.e. $\geq 4.0\%D$). For small displacement, the smaller the displacement is, the closer the lateral resistance ratio is to the relative stiffness; for large displacement, the larger the displacement, the closer the lateral resistance ratio is to the relative strength.

Figures 501.40, 501.41 and 501.42 summarize observed lateral resistance ratios in layered profiles. Figure 501.40 shows the lateral resistance ratios in the intermediate sand layer corresponding to various relative stiffness $R_{stiffness}$ at pile displacement of 0.5% D for the Clay–Sand–Clay model. Figures 501.41 and 501.42 show the lateral resistance ratios corresponding to various relative stiffness $R_{stiffness}$ and relative strength $R_{strength-FEM}$ at pile displacements of 0.5% D and 6.5% D for the Sand–Clay–Sand model. The effects of the intermediate clay layer on the upper sand layer reduce to less than 10% at a distance of 0.5 to 1.5 D above the interface at small pile displacement (e.g. 0.5% D), while that effects reduce to less than 10% at a distance of 1.25 to 2.0 D above the interface at large pile displacement (e.g. 6.5% D).

One may notice that the lateral resistance ratios corresponding to the relative stiffness $R_{stiffness} = 0.63$ in Figures 501.40 and 501.41 are not the same. The ratios close to the lower sand interface in the Clay–Sand–Clay model is slightly larger than that in the Sand–Clay–Sand model. This difference is due to the fact that the lateral resistance ratios in the intermediate sand layer also include the effects of smaller unit weight of upper layer clay.

501.3.6 Summary

This subsection summarizes results from finite element analysis on the behavior of a single pile in elastic–plastic layered soils. Based on the results presented, the following conclusions can be drawn.

1. The layering effects are two–way. Not only the lower layers are affected by the upper layers, but the upper layers are also affected by the lower layers. Furthermore, the layering effects are not symmetric. In the case of pile laterally loaded at the pile head, the effect of an interface extends further into the layer above the interface than it does into the layer below the interface at small displacements.

2. In the Clay–Sand–Clay model, the lateral resistance of soft clay increases by as much as 30% and the effect extends to one pile width above the upper sand interface for $R_{stiffness} = 0.63$ at small pile displacement (0.5% D). Nonetheless, the increase of lateral resistance in the upper clay layer at large pile displacement (8–10% D) extends only one finite element above the upper sand interface.
Figure 501.40: Summary of observed lateral resistance ratios from FEM analysis for the Clay–Sand–Clay profile at small deflection (y/D=0.5%).

Figure 501.41: Summary of observed lateral resistance ratios from FEM analysis for the Sand–Clay–Sand profile at small deflection (y/D=0.5%).
On the other hand, the clay layers also have significant effects on the lateral resistance of sand throughout the intermediate layer.

3. In the Sand–Clay–Sand model, the intermediate clay layer has considerable effects on the lateral resistance of the upper sand layer, and the sand layers also have significant effects on the lateral resistance of the intermediate clay layer, causing 10 to 40% increase in its lateral resistance.

4. The lateral resistance ratio is dominated by the relative stiffness at small displacements (i.e. ≤ 1.0%D), while that is controlled by the relative strength at large displacements (i.e. ≥ 5.0%D).

It must be pointed out that the above observed lateral resistance ratios may only be applied to similar stratigographies, pile deformation modes, and other conditions considered in this work. Further analyses are needed to investigate the effects of other stratigographies, pile deformation modes, pile diameters, and other factors, in order to draw more general guidelines. Future studies with a refined mesh around the interface will provide better resolution of the resistance ratio around the interface. Future studies of the effects of the interface layer on the layering effects will also be very interesting.
501.4 Numerical Study of Group Effects for Pile Groups in Sands

Material presented here has been previously published in our paper Yang and Jeremić (2005).

501.4.1 Introduction

Single pile foundations have been successfully modeled by the $p-y$ approach, as implemented in LPILE (Reese et al. Reese et al. (2000a)). However, the behavior of a pile within a group may differ greatly from that of a single pile and vary depending on the position due to the interaction between the neighboring piles. To study this interaction effects, only a couple of field tests have been carried out because of the large costs incurred. Brown et al. Brown et al. (1988) conducted cyclic loading tests on instrumented $3 \times 3$ steel pile group. The $p$–multiplier concept was presented based on the measured soil resistance data and specific $p$–multipliers were suggested for the three rows. Ruesta and Townsend Ruesta and Townsend (1997) reported an in-situ test on piles at Roosevelt Bridge. Rollins et al. Rollins et al. (1997) tested another full–scale pile group founded in clay and suggested a set of $p$–multipliers for corresponding pile groups. Ng et al. Ng et al. (2001) presented results on full–scale lateral load tests of one single pile and three pile groups with large–diameter bored piles. Besides in-situ testing, many centrifuge tests were conducted to predict the behaviors of pile groups under static and dynamic loading. Recently, McVay et al. McVay et al. (1995) McVay et al. (1998) conducted a series of lateral load tests on large pile groups ($3 \times 3$ to $7 \times 3$) founded in sands to study the interaction effects within a group.

Based on these field and centrifuge tests, it was found that:

1. All the test results have clearly shown that the lateral resistance of a pile within the group is strongly influenced by its row position and the $p$–multiplier method was suggested by Brown et al. Brown et al. (1988) to account for this behavior. Specifically, each row within the group is assigned a different $p$–multiplier $f_m$ and the $p$–$y$ curve for a single pile is multiplied by $f_m$ to produce $p$–$y$ curves for all the piles in the same row.

2. The $p$–multipliers are independent of soil density and only depend on the pile geometry. And at sufficient deformation and under static loading, they are constant for practical purposes (McVay et al. McVay et al. (1998)).

3. For 3–diameter spacing, the suggested $p$–multiplier $f_m$ was 0.8, 0.4, 0.3 (front row to back row) for the $3 \times 3$ group (Brown et al. Brown et al. (1988)), 0.8, 0.4, 0.3, 0.3 for the $4 \times 3$ group, and 0.8, 0.4, 0.3, 0.2, 0.3 for the $5 \times 3$ group, and 0.8, 0.4, 0.3, 0.2, ..., 0.3 for all larger group size (McVay et al. McVay et al. (1998)).
4. For concentric loading (located at the geometrical center of the pile group), the difference between the side and middle piles within a row is small and may be neglected, while the moments in the side piles within a given row are slightly larger than that in the middle piles but may be represented by the average (McVay et al. McVay et al. (1998)).

Together with the physical modeling, a few numerical simulations have also been performed. We mention a few representative finite element studies of pile groups. Maqtadir and Desai Muqtadir and Desai (1986) studied the behavior of a pile-group using a three dimensional program with nonlinear elastic soil model. An axisymmetric model with elastic-perfectly plastic soil was used by Pressley and Poulos Pressley and Poulos (1986) to study group effects. Brown and Shie Brown and Shie (1990a) Brown and Shie (1990b) Brown and Shie (1991) and Trochanis Trochanis et al. (1991) conducted a series of 3D Finite Element Method (FEM) studies on the behavior of single pile and pile group with elastic-plastic soil model. In particular, interface element was used to account for pile-soil separation and slippage. Moreover, several model and field tests of free- or fixed-head pile groups have been analyzed by Kimura et al. Kimura et al. (1995) and Wakai et al. Wakai et al. (1999) using 3D elasto-plastic FEM.

This paper describes 3D elastic-plastic finite element modeling of two pile groups founded in sands with emphasis on the interaction effects within pile group. Specifically, bending moment and load distribution in individual piles were examined and compared with centrifuge test data. Special attention was given to out-of-loading-plane bending moment and \( p-y \) behavior of individual piles in a group. The OpenSees OpenSees Development Team (Open Source Project) (2000-2006) finite element framework was employed to complete all the computations. Soil modeling was performed using the Template Elastic–Plastic approach (Jeremić and Yang Jeremić and Yang (2002)).

This paper is organized as follows. Section 501.4.2 summarizes the centrifuge tests and describes finite element models including the soil elastic–plastic model used for 3×3 and 4×3 pile group simulations. Section 501.4.5 presents a number of results and discussion describing simulated behavior of analyzed pile groups. In particular, presented are developed plastic zones (Section 501.4.6), pile bending moments (Section 501.4.7), pile load distributions (Section 501.4.8), comparison of \( p-y \) curves for individual piles (Section 501.4.9), and comparison with centrifuge tests (Section 501.4.10). Section 501.4.11 gives concluding remarks.
501.4.2 Pile Models

501.4.3 Summary of Centrifuge Tests

In the centrifuge tests reported by McVay et al. McVay et al. (1998), 3 × 3 to 7 × 3 pile groups embedded in homogeneous sands under lateral load were tested. The rectangular–shaped sample container was fabricated from aluminum alloy with an inside dimensions of 0.254 m wide, 0.457 m long, and 0.305 m high. The piles were spaced by three times the pile width and the pile caps (made of aluminum) were rigidly connected with the piles. The model square piles and pile cap were fabricated from solid square aluminum (alloy 6061) bars. Each individual pile is 9.5 mm wide and 304.8 mm long. To simulate the installing effects of field driven piles, the piles were driven in flight into sands by hydraulic equipment and tested at 45 g.

The sands (artificially mixed by a number of different gradations) studied were at two different relative densities: a loose sand with relative density $Dr = 36\%$, unit weight $\gamma = 14.05 \text{kN/m}^3$ and a medium dense sand with $Dr = 55\%$, $\gamma = 14.50 \text{kN/m}^3$. The shear modulus $G$, Poisson’s ratio $\nu$ and friction angle $\phi$ are 8230 $\text{kN/m}^2$, 0.35, 34.5° for the loose sand and 8960 $\text{kN/m}^2$, 0.35, 37.1° for the medium dense sand. It is noted that the friction angles were determined from drained triaxial compression tests, and the shear moduli were back–computed from instrumented vertical load tests (Zhang et al. Zhang et al. (1998)) and were valued at a depth of 13.7 m.

501.4.4 Finite Element Pile Models

Among these tested pile groups, 3 × 3 and 4 × 3 groups were chosen to be modeled in prototype scale using 3D elasto–plastic finite element method to investigate pile group interaction effects. The typical layout of 4 × 3 pile group is shown in Fig. 501.43. The whole centrifuge model in prototype scale is 22.8 m wide, 20.6 m long and 13.2 m deep. Only half of each centrifuge model is meshed considering the symmetry. Figure 501.44 shows the finite element mesh for the 4 × 3 pile group. Additional finite element analysis of a cantilever beam using the same mesh as an individual pile in the group was carried out and comparison of the displacement at the top of the beam from FEM and beam theory solution indicated that the mesh was fine enough to capture the pile behavior. Soil, pile and soil–pile interface are all modeled with twenty node brick elements. Each pile consists of four elements (per cross section) made of elastic material with properties corresponding to aluminum. There are 1268 and 1414 brick elements in the two models respectively. The sides and bottom of each model are fixed in all three coordinate directions with the exception of the symmetric boundary, which is only supported in the direction perpendicular to the symmetry plane. This type of boundary conditions are fairly close to the actual friction boundary conditions in the centrifuge tests.
All the parameters except the Young’s moduli for sands were the same as from the centrifuge studies. The Young’s modulus is assumed to depend on the mean effective normal stress $p’$ (Manzari and Dafalias Manzari and Dafalias (1997)) as:

$$E = E_o \left(\frac{p’}{p_a}\right)^n$$  \hspace{1cm} (501.5)

where $E_o$ is Young’s modulus at the atmospheric pressure, $p’ = \sigma_{ii}/3$ is the mean effective normal stress, $p_a$ is the atmospheric pressure, and $n$ is constant for a given void ratio. Usually 0.5 is used for $n$. For the medium dense sand, the Young’s modulus at the atmospheric pressure computed by Eqn. (501.5) (Lateral pressure coefficient $K_o = 0.55$ was used) from the back–computed shear modulus was 200000 kPa and then adjusted to be 17400 kPa for the medium dense sand, so as to well simulate the load–displacement curve obtained from centrifuge tests. For the loose sand, the computed Young’s modulus at the atmospheric pressure from the back–computed shear modulus was 18700 kPa and similarly adjusted to be 16000 kPa.

Sand was simulated by Drucker–Prager material model with nonassociated flow rules. Since the centrifuge studies we used to compare our simulations against did specify only the friction angle of test sands which were obtained from drained triaxial compression tests, the yield surface was chosen to agree with Mohr–Coulomb hexagon at triaxial compression. Future study using Mohr–Coulomb material model will be useful to determine the effects of varying friction angle in Drucker–Prager model on the results. Since there is no test data on the dilation angle $\psi$ of the tested sands, a dilation angle of 0° was used in this work, as similar dilation angle was also used in Brown and Shie Brown and Shie (1990a). The soil–pile interface was represented by one thin layer of elements. The material of the interface element was also simulated by Drucker–Prager model with a friction angle of 25°, a dilation angle of 0°, and the same Young’s modulus and Poisson’s ratio as corresponding sands. In the future, a realistic dilation angle needs to be used for sand to further investigate the effects of dilation angle on the pile group interaction behavior, especially the out–of–loading–plane bending moment.

501.4.5 Simulation Results

In this section we present results related to the behavior of $3 \times 3$ and $4 \times 4$ pile groups in loose and medium dense sands. A number of static pushover tests were simulated using FEM. Specifically, modified Newton–Raphson method were used to solve the system of equations in the finite element level and implicit algorithm (Jeremić and Yang Jeremić and Yang (2002)) was used in constitutive level integration. Results are also compared with those from the centrifuge studies by McVay et al. McVay et al. (1998) and Zhang et al. Zhang et al. (1999).
501.4.6 Plastic Zone

The static pushover tests were conducted using load control at the pile head with the loading applied in the X direction. The final plastic zones (represented by plastified Gauss points) for two pile groups are depicted in Figures 501.45 and 501.46. In particular, Figure 501.45 shows different views of the 3D plastic zone developed in the $3 \times 3$ pile group at the lateral load of 2,200 kN. Figure 501.46 shows the different views of the 3D plastic zone developed in the $4 \times 3$ pile group at the lateral load of 2,970 kN.

---

Figure 501.43: Layout of $4 \times 3$ pile group: a) top view, b) side view.

Figure 501.44: Finite element mesh for half of $4 \times 3$ pile group.
kN. Both figures clearly show a wedge shaped plastic zone at the shallow depth. Also apparent is the propagation of the plastic zone (shear yielding) along the pile–soil interface, resulting from the rocking behavior of the group.

Figure 501.45: Plastic Gauss–Points for the $3 \times 3$ pile group: (a) 3D view and (b) side view.

Figure 501.46: Plastic Gauss–Points for the $4 \times 3$ pile group: (a) 3D view and (b) side view.
501.4.7 Bending Moment

The maximum bending moment in the piles often controls the design of pile groups and therefore has to be analyzed accurately. In order to generate the bending moments from the stress field in piles vertical stresses at Gauss points from each pile element are integrated numerically. The moments with reference to the Y-axis ($M_y$) of each individual pile in the $3 \times 3$ and $4 \times 3$ pile groups are plotted in Figure 501.47 and 501.48, respectively. Figure 501.47 shows the moment diagrams of the $3 \times 3$ group at a lateral load of 2200 kN. The calculated maximum moment for each pile occurs at the pile cap, which is consistent with the fixed pile cap condition. The lead row piles, for both side and middle piles, in both $3 \times 3$ and $4 \times 3$ pile groups, carry the maximum bending moment. It is also interesting to note that in the $4 \times 3$ group, the moment diagrams for the piles in the third and fourth rows appear to be almost identical, which implies that they behave almost the same within the group.

The variations of maximum bending moments in each pile of $3 \times 3$ and $4 \times 3$ groups are illustrated in Figures 501.49 and 501.50, respectively. The maximum moments develop in the lead–row side piles, while the smallest maximum moments occur in the trail–row middle piles in both groups. It is obvious that the maximum moments developed in the middle and side piles within the lead row are quite different for both pile groups, implying the load shared by each pile in the same row is different. For example, in the $3 \times 3$ group, the maximum moment in the middle pile at the end load is 600 kN.m, while that of the side pile is 670 kN.m, the difference is about 11%. For the $4 \times 3$ pile group, the maximum moment on the lead–row side pile was about 10% greater than that for the lead–row middle pile.

It is interesting to look at the moments with reference to the X-axis (out-of-loading-plane moment, $M_x$). Figure 501.51 (a, b) shows $M_x$ diagram for each pile as well as deformed piles for the $3 \times 3$ pile group. Similar plots for the $4 \times 3$ pile group are shown in Figure 501.53 (a, b). The maximum value of $M_x$ in the $3 \times 3$ pile group reaches 50 kN.m, which is about 8% of the maximum value of $M_y$. For the $4 \times 3$ pile group, the maximum value of $M_x$ is about the same amount, which is about 6% of the maximum value of $M_y$.

It is noted that the signs of the $M_x$ moments in the lead and trail rows are different, indicating the bending directions are opposite to each other. This is further verified by looking at the deformed shape of the pile group shown in Figure 501.51(b) with only the displacement in Y direction shown. This kind of bending is caused by the complex displacement field of the soil surrounding the pile group, as illustrated by the horizontal displacement vector and contour plots of displacement in X direction in Figures 501.52 and 501.54 for the $3 \times 3$ and $4 \times 3$ pile groups, respectively. The soil in front of the lead row tends to “squeeze into” the group, while the soil outside of the trail row tends to “come back” toward the pile group when the pile group is moving forward, which consequently results in the fact that
Figure 501.47: Comparison of bending moment diagram at lateral load of 2200 kN for piles in $3 \times 3$ group.

Figure 501.48: Comparison of bending moment diagram at lateral load of 2970 kN for piles in $4 \times 3$ group.
Figure 501.49: Maximum bending moments in individual piles in $3 \times 3$ group.

Figure 501.50: Maximum bending moments in individual piles in $4 \times 3$ group.
the lead row bends outward and the trail row bends inward.

Figure 501.51: (a) Out-of-loading-plane bending moment diagram and (a) Out-of-loading-plane deformation for the $3 \times 3$ pile group.

Figure 501.52: Horizontal displacement vector and contour of displacement in X direction at ground surface for the $3 \times 3$ pile group.
Figure 501.53: (a) Out-of-loading-plane bending moment diagram and (b) Out-of-loading-plane deformation for the $4 \times 3$ pile group.

Figure 501.54: Horizontal displacement vector and contour of displacement (cm) in X direction at ground surface for $4 \times 3$ pile group.
501.4.8 Load Distribution

In order to compute the load taken by each pile, the values of bending moment at element centers along with the boundary condition at the bottom of pile (zero moment) were fitted with a 5th order polynomial by least square technique. According to the theory of beam on a Winkler-type subgrade (Hartog 1952), the moment curve was differentiated once to compute the shear force. Then, the shear forces at three sampling points between the ground surface and the pile cap were averaged to compute the load carried by each individual pile. The accuracy of the load measuring scheme has been verified by comparing the total load actually applied on the pile cap and the sum of all loads carried by each individual pile.

![Lateral Load Distribution in Each Row (kN)](image)

Figure 501.55: Variations of load taken by each row in the $3 \times 3$ group.

Figures 501.55 and 501.56 show the load and percentage of total load carried by each row of the $3 \times 3$ group. It is easily seen that not only the load but the percentage of total load taken by each row, especially the lead and trail rows, change steadily during loading process. It is observed in Figure 501.55 that the lead row and trail row share almost the same amount of load at small lateral displacement. However, as deflection increases, the lead row picks up the load much faster that the trail row, although the load taken by the three rows all increase.

More interestingly, the variation in the percentage of load carried by each row exhibits completely different trends. In the initial loading stage, the percentage shared by the lead row increases and that by the trail row drops quickly, while the percentage shared by the middle row almost remains constant.
In the final loading stage, however, the percentages carried by the lead and trail rows tend to stabilize. The lead row takes the most load, more than 46%, while the trail row takes the least, only around 25%.

In addition, Figure 501.56 shows that the trail row takes a greater percentage of load than the lead row at small lateral deformation, which were attributed to some load measuring error and the fact that the denominator is relatively small.

Similar plots for the $4 \times 3$ group are shown in Figures 501.57 and 501.58. While the lead row still carries much more load that the trail row, the third row and the trail row share almost the same amount of load at large lateral displacement, which is in agreement with the fact that the same $p$-multiplier was recommended for the third and fourth row by McVay et al. McVay et al. (1998). It should also be observed from Figures 501.57 and 501.58 that the lead row carries more than twice the load of the trail row.

It is worthwhile noting that the distribution of load in the same row can be quite different. Figures 501.59 and 501.60 show the variations of load and the percentage of total load taken by each pile of the $3 \times 3$ group, respectively. Similar plots for the $4 \times 3$ group are shown in Figures 501.61 and 501.62. It is obvious that the piles at the sides take more load than the piles in the middle at the same row. For the $3 \times 3$ group, the side pile in lead row takes 350 kN or 16% of total load while the middle pile takes 325 kN or 14.6% of total load at the end of loading.
Figure 501.57: Variations of load taken by each row in the $4 \times 3$ group.

Figure 501.58: Variations of percentage of total load taken by each row in the $4 \times 3$ group.
Figure 501.59: Variations of load taken by each pile in the $3 \times 3$ group.

Figure 501.60: Variations of percentage of total load taken by each pile in the $3 \times 3$ group.
Figure 501.61: Variations of load taken by each pile in the $4 \times 3$ group.

Figure 501.62: Variations of percentage of total load taken by each pile in the $4 \times 3$ group.
501.4.9  \( p - y \) Curve

To further investigate the behavior of each pile, it is worthwhile to examine the \( p - y \) behavior of each pile in the group. The fitted moment curve and the resulting displacement from FEM were used to derive the \( p - y \) curves according to beam theory. Figures 501.63, 501.64, 501.65 and 501.66 show the derived \( p - y \) curves at given depths for all individual piles in the two groups. As is evident from \( p - y \) plots for the two groups, the piles in the lead row exhibit much larger resistances than the piles in the middle and trail rows at large lateral displacement, due to the well-known “shadowing effect”. Comparison of the \( p - y \) curves at different depths in the same pile shows that the lateral resistance \( p \) increases as depth increases. For example, at deflection of 4 cm in the 3D (3 pile diameters) spaced \( 3 \times 3 \) group, the lateral resistance on the lead-row side pile is 120 kN/m at a depth of -1.54 m, while it is only 90 kN/m at a depth of -0.58 m. This is caused by the increases in vertical stress and Young’s modulus as depth increases.

More interestingly, it is seen that the \( p-y \) curves of the lead–row and trail–row piles at the depth of -1.54 m are identical at small lateral displacement \( (y < 0.5 \text{ cm}) \). Then the \( p-y \) curves of the piles within the trail row soften drastically as lateral displacement increases in the \( 3 \times 3 \) group. Similarly for the \( 4 \times 3 \) group, both the \( p-y \) curves of the lead–row and trail–row piles, and the \( p-y \) curves of the third–row and second–row piles at the depth of -2.04 m are almost identical at small lateral displacement \( (y < 0.7 \text{ cm}) \). As deflection increases, the \( p-y \) curves of the trail–row and third–row piles soften quickly.

Obviously, each pile in the group exhibits quite different behavior than each other. It is believed that the different behavior of each pile is directly related to the yielding of soil in front of these piles. This observation can be verified by the fact that the softening behavior of the trail-row piles starts at larger lateral displacement as the depth increases, since the plastic zone first develops at the ground surface and then extends downward as deflection increases. These observations also imply that the \( p \)-multiplier approach might not be appropriate, especially at small deflection, since it obtains the \( p - y \) curves for piles within the group by simply scaling the single–pile \( p - y \) curve. In addition, as a comparison with 3D (3 pile diameters) spaced pile group, also plotted in Figures 501.63 and 501.64 is the \( p-y \) curve of a lead–row side pile from a 6D spaced pile group. It is apparent that the lateral resistance increases with the increase of spacing.

501.4.10  Comparison with the Centrifuge Tests

The pile head displacements for the two pile groups from 3D FEM and centrifuge tests (McVay et al. McVay et al. (1998)) were plotted against pile head load in Figures 501.67 and 501.68. It can be seen that they agree with each other fairly well at the small lateral displacement and the FEM model is
relatively stiffer at large lateral displacement.

The accuracy of finite element modeling can also be examined in terms of maximum bending moments. The maximum moment developed in $4 \times 3$ group was compared with that from the centrifuge study (Zhang et al. (1999)) in Figure 501.69. The results from the centrifuge study are slightly larger than that from FEM, which is partially due to the relatively simple elastic-plastic soil model used. Moreover, all the above results can be further improved if the mesh is refined, since the refined pile group mesh will exhibit softer load–displacement response, and therefore develop larger bending moment in individual piles.

The percentage of total lateral load taken by each row from 3D FEM and centrifuge tests (McVay
et al. McVay et al. (1998)) at a lateral load of 1650 kN and 2300 kN for the $3 \times 3$ and $4 \times 3$ pile groups, respectively, were compared in Figures 501.70 and 501.71. Results for both loose and medium dense sand cases are included. Figures 501.70 and 501.71 show that the density of sand does not have much effects on the load distribution. It is evident that the load distributions to all rows as obtained from FEM and centrifuge tests in all the cases for both pile groups agree very well (the differences are within 3%).

Finally, the variation of distribution of total load to each row as obtained from 3D FEM and centrifuge tests for the $4 \times 3$ pile group was compared in Figure 501.72. At small lateral displacement, the FEM
Figure 501.67: Comparison of load displacement response for the $3 \times 3$ pile group.

Figure 501.68: Comparison of load displacement response for the $4 \times 3$ pile group.
Figure 501.69: Comparison of maximum bending moment response for the $4 \times 3$ group.

Figure 501.70: Comparison of percentage of total lateral load taken by each row in the $3 \times 3$ group.
Figure 501.71: Comparison of percentage of total lateral load taken by each row in the 4 × 3 group

computed percentage of total load acting on the lead row was smaller than measured in the centrifuge tests and the load acting on the trail row was larger than measured in the centrifuge tests. When lateral displacement is beyond 3–4 cm, however, the FEM computed load distribution tends to stabilize and agrees well with that from centrifuge tests.
Figure 501.72: Variation of percentage of total load taken by each row in the 4 × 3 pile group.
501.4.11 Conclusions

This paper presents results from the finite element study on the interaction effects of pile groups founded in sands. Specifically the $3 \times 3$ and $4 \times 3$ pile groups were analyzed in terms of plastic zone, bending moment and load distribution among individual piles. Comparison of results from FEM and centrifuge study shows that elasto–plastic finite element analysis can predict the behavior of pile group with very good accuracy. Particularly, load distribution results from finite element analyses agree very well with that from centrifuge study.

It was shown that not only the load taken by each row in the group is different, but the load shared by individual piles and maximum bending moment developed in individual piles within the same row vary quite a bit, as observed in the centrifuge tests. Although the difference between the loads taken by lead–row middle and side piles is less than 2% of the total load, the difference between the maximum bending moments developed in the lead–row middle and side piles reaches 10–11% in the two pile groups. Interestingly, it was found that bending moment also occurs in the plane perpendicular to the loading direction.

The numerically generated $p-y$ curves were used to study the behavior of each pile in a group. It was found that individual piles in the group exhibit quite different $p-y$ behavior at small deflection, which means that the interaction in elastic range is different than that for loading in plastic range. Therefore, different interaction factors would be more appropriate depending on the loading range.

Since FEM can capture the critical aspects of group effects, it could now be used to systematically study various pile group configurations at much smaller cost than actual load tests, and derive interaction factors for elastic and plastic loading levels that could be used in standard design practice.
Chapter 502
Earthquake Soil Structure Interaction, General Aspects


(In collaboration with Dr. Nima Tafazzoli, Prof. José Abell, Dr. Yuan Feng, Dr. Han Yang, and Dr. Hexiang Wang)
502.1 Chapter Summary and Highlights

502.2 Free Field Ground Motions

Seismic waves propagate from the source (rupturing fault) through the bedrock, surface rock and soil layers to the site of interest where a Nuclear Power Plant is located (or planned). Seismic (compressional and shear) waves do travel through various rock and soil domains, which can be represented by layers, which are sometimes horizontal, but mostly inclined. In addition to that, layers usually have variable thicknesses, creating a complex underground picture of soil/rock domains of different stiffness, mass, energy dissipating characteristics (damping). Most commonly, the stiffness of layers (horizontal and/or inclined) increases with depth. This change in stiffness results in seismic wave refraction, as shown in Figure 502.1.

![Figure 502.1: Propagation of seismic waves in nearly horizontal local geology, with stiffness of soil/rock layers increasing with depth, and refraction of waves toward the vertical direction.](image)

When soil/rock layers are mostly horizontal, inter-layer refraction causes seismic waves (both P and S) to tend toward vertical propagation. This case is shown in Figure 502.1. However, (near) horizontal layering of geologic layers is not very frequent, unless young deposits of soil in river valleys are of concern. More often the soil/rock layers are inclined, thus creating conditions for variable directivity of seismic motions due to refraction. Figure 502.2 shows one such case where inclined soil/rock layers contribute to mainly horizontal propagation of seismic motions close to the surface, in the vicinity of an NPP.

A general conclusion can be made that seismic waves arriving at the particular site (surface) will be fully three dimensional (3C), uncorrelated, and incoming at an oblique angle. As noted by Zerva (2009), seismic motions will feature lack of correlations of motions between two monitored points at the surface. Lack of correlation is mainly due to (see more details in section 109.2.8):

- Attenuation effects,
Figure 502.2: Propagation of seismic waves in inclined (close to vertical) local geology, with stiffness of soil/rock layers increasing through geologic layers, and refraction of waves away from the vertical direction.

- Wave passage effects,
- Scattering effects,
- Extended source effects

Due to all the above mentioned 3C effects, realistic modeling of seismic wave propagation can only properly be done by fully and realistically incorporating:

- Body waves (P and S),
- Surface waves (Love and Rayleigh),
- Seismic waves coming at an oblique angle (which will actually be done when the above two types of waves are properly modeled),
- Full three dimensional (3C) wave field,
- Lack of correlation (incoherence) effects,
- Energy dissipation effects (damping).
502.2.1 Seismic Motions: Available Data

There exist a large number of recorded earthquake motions. Most records feature data in three perpendicular directions, East-West (E-W), North-South (N-S) and Up-Down (U-D). Number of recorded strong motions, is (much) smaller. A number of strong motion databases (publicly available) exist, mainly in the east and south of Asia, west cost of north and south America, and Europe. There are regions of world that are not well covered with recording stations. These same regions are also seismically fairly inactive. However, in some of those regions, return periods of (large) earthquakes are long, and recording of even small events would greatly help gain knowledge about tectonic activity and geology.

Ergodic Assumption. Development of models for predicting seismic motions based on empirical evidence (recorded motions) rely on Ergodic assumption. Ergodic assumption allows statistical data (earthquake recordings) obtained at one (or few) worldwide location(s), over a long period of time, to be used at other specific locations at certain times. This assumption allows for exchange of average of process parameters over statistical ensemble (many realizations, as in many recordings of earthquakes) is the same as an average of process parameters over time.

While ergodic assumptions is frequently used, there are issues that need to be addressed when it is applied to earthquake motion records. For example, earthquake records from different geological settings are used to develop GMP equations for specific geologic settings (again, different from those where recordings were made) at the location of interest.

3C (6C) versus 1C Records/Motions. Recordings of earthquakes around the world show that earthquakes are almost always featuring all three components (E-W, N-S, U-D). There are very few known recorded events where one of the components was not present or is present in much smaller magnitude. Presence of two horizontal components (E-W, N-S) of similar amplitude and appearing at about the same time is quite expectable. The four cardinal directions (North, East, South and West) which humans use to orient recorded motions have little to do with the earthquakes mechanics. The third direction, Up-Down is different. Presence of the vertical motions before main horizontal motions appear signify arrival of Primary (P) waves (hence the name). In addition, presence of vertical motions at about the same time when horizontal motions appear, signifies Rayleigh surface waves. On the other hand, lack (or very reduced amplitude) of vertical motions at about the same time when horizontal motions are present signifies that Rayleigh surface waves are not present. This is a very rare event, that the combination of source, path and local site conditions produce a plane shear (S) waves that surfaces (almost) vertically. One such example (again, very rare) is a recording LSST07 from Lotung recording array in Taiwan (Tseng et al., 1991). Figure 502.3 shows three directional recording of earthquake LSST07 that
occurred on May 20th, 1986, at the SMART-1 Array at Lotung, Taiwan.

![Acceleration time history LSST07 recorded at SMART-1 Array at Lotung, Taiwan, on May 20th, 1986. This recording was at location FA25. Note the (almost complete) absence of vertical motions, signifying absence of Rayleigh waves. Figure from Tseng et al. (1991).](image)

Figure 502.3: Acceleration time history LSST07 recorded at SMART-1 Array at Lotung, Taiwan, on May 20th, 1986. This recording was at location FA25. Note the (almost complete) absence of vertical motions, signifying absence of Rayleigh waves. Figure from Tseng et al. (1991).

Note almost complete lack of vertical motions at around the time of occurrence of two components of horizontal motions, signifies absence of Rayleigh surface waves. In other words, a plane shear wave front was propagating vertically and surfaced as a plane shear wave front. Other recordings, at locations FA15 and FA35 for event LSST07 reveal almost identical earthquake shear wave front surfacing at the same time (Tseng et al., 1991).

On the other hand, recording at the very same location, for a different earthquake (different source, different path) (LSST12, occurring on July 30th 1986) reveals quite different wave field at the surface, as shown in Figure 502.4.
Figure 502.4: Acceleration time history LSST12 recorded at SMART-1 Array at Lotung, Taiwan, on July 30th, 1986. This recording was at location FA25. Figure from Tseng et al. (1991).
502.2.2 Multi-Directional and Seismic Input Coming in at Inclined Angle

Both multi-directional and seismic input coming at an inclined angle is possible by using (and in fact it is inherent to) the DRM, described in previous section. It is important to note that both seismic body waves (P and S) and seismic surface waves (Rayleigh and Love) are present in all situations, and do contribute to multi-directional and inclined seismic input.

![Illustration of seismic body waves, namely the P (primary) and S (secondary) waves](image)

Figure 502.5: Illustration of seismic body waves, namely the P (primary) and S (secondary) waves (illustration from MTU web site).

**Importance of surface waves.** It has long been recognized that surface waves are responsible for majority of destruction and the seismic energy at some distance from the epicenter during earthquakes (Kramer (1996a), Semblat and Pecker (2009)). In a perfectly linear elastic half-space, Rayleigh waves become significant source of seismic motions at distance $R = h \sqrt{v_p/v_R} / (v_p/v_R)^2 - 1$ where $h$ is hypocenter depth (for uniform half-space) and $v_p$ is the compressional (primary) wave velocity, while $v_R$ is the Rayleigh wave velocity (Kramer (1996a)). For example, if $v_R \approx 0.93v_S = 0.93 \times 0.6v_p = 0.56v_p$ it follows that $R \approx 1.5h$ where $v_S$ is the shear wave velocity (Semblat and Pecker (2009)). Of course for a case with realistic geology, the pattern of occurrence of Rayleigh waves is much more complex. However, since Rayleigh waves represent the result of interaction of body waves (P and S) with the free surface, it is safe to conclude that they are always present.
502.2.3 Free Field Motion Development

For verification of and for full analysis of ESSI using the DRM, we develop seismic motions using a number of approaches

- Closed form solution,
  - One Component (1C) of three times one components ($3 \times 1C$) using deconvolution (recommended) or convolution (not recommended, due to possible and negative interference of seismic waves at depth, see examples with the SMR)
  - Three Components (3C), plane waves, body and surface (Thomson, 1950; Haskell, 1953)

- Integration equation, Green’s functions (frequency – wavenumber method, fk)

- FEM (fault slip model using Real-ESSI)

- Finite Difference models, SW4

502.2.3.1 Details of Free Field Motion Development

**Closed Form Solution** Analytic solutions used can impose simplified motions (not all components):

- Monochromatic harmonic wave
- Body wave: P, SV, SH
- Surface wave: Rayleigh, Love
- Incident, reflected, transmitted wave components

- Ormsby wavelet
  - Body wave: P, SV, SH
  - Surface wave: Rayleigh, Love
  - Incident, reflected, transmitted wave components

- Ricker wavelet
  - Body wave: P, SV, SH
  - Surface wave: Rayleigh, Love
  - Incident, reflected, transmitted wave components

Definitions for Ormsby and Ricker wavelet are given in chapter 310.2 on page 1721.

Closed Form Solution – Ormsby Wavelet?

- Ormsby wavelet is a broad band signal
- Idea: "Shake" the model (or components) with a broad frequency signal to evaluate effects of different frequencies
- Verifying models, propagation of waves of different frequencies
Closed Form Solution – Ricker Wavelet?

- Ricker wavelet is a narrow band signal
- Idea: "Shake" the model (or components) with a narrow frequency signal to evaluate its effects
- Verifying models, propagation of waves of this particular frequency

Frequency-Wavenumber Method (fk method)

- fk method developed originally by Haskell (1964), Wang and Herrmann (1980)
- Based on Green’s functions
- Current fk program developed by Zhu and Rivera (2002)
- available at http://www.eas.slu.edu/People/LZhu
- fk program is widely used in seismology and geophysics

Fault Slip Model (FEM)

- Develop large FEM model (5km × 10km)
- One element (or more) used to initiate the fault rupture
- Fault rupture can be a realistic stress drop, or Ormsby or Ricker wavelets
- Use initial set of motions before the reflections of boundaries interfere (but can use those too!)

FE Model
Figure 502.9: 2D model.

Figure 502.10: 2D model, with observation point location.

**FE Model**

**Properties**

- Model Properties
  - $190 \text{ m} \times 100 \text{ m} \times 10 \text{ m}$ dimension
  - $V_s = 100 \text{ m/s}$
- Poisson’s ratio = 0.3
- Density = 2000 kg/m³

• Input Wave Properties
  - Ricker Wavelet (dominant frequency 1 Hz)
  - Body (SH, SV), Surface (Rayleigh)
  - 0°, 30° inclination from the vertical
  - Evaluated closed-form solution (incident, reflected waves)

![Figure 502.11: Results at the observation point, vertical incident SV wave, horizontal displacements.](image)

**SH Ricker Wavelet Input, 0 deg, y Component**

**502.2.4 Free Field Ground Motions Development: Closed Form Solution**

**502.2.4.1 Three Component, 3C Motion Development**

Thomson (1950), Haskell (1953)

**502.2.4.2 One Component, 1C Motion Development**

**Note on Viscous Damping for 1C Deconvolution and Convolution.** Viscous damping used in 1C analytic solution needs to be properly calibrated. For deconvolution, using analytic/closed form solution, propagation of surface record back in time to the depth the viscous damping model used is Voigt damping. Voigt viscous damping features constant damping ratio over all frequencies. For propagation of that same wave back to the surface using FEM solution, Rayleigh viscous damping is used. Rayleigh viscous damping is not constant over frequencies. Rayleigh damping parameters $\alpha$ and $\beta$ ($C = \alpha M + \beta K$) have to be calibrated. Calibration of Rayleigh viscous damping parameters $\alpha$ and $\beta$ has to be done in
such a way so that Rayleigh viscous damping can approximate behavior of Voigt viscous damping over main frequency range of seismic motion.

It is suggested that for deconvolution and convolution, propagation of motions upward, damping parameters first be set as zero (0.0) for both de-convolution and FEM propagation. This is done to verify the model and make sure that everything else is correct and that surface free field response can be recovered. This is important in order to make sure that surface motions can be successfully recovered from prescribed motions, through deconvolution and convolution process, that rely on different methods, and without influence of damping.

Only after model is verified with no viscous damping, one should proceed to calibrate viscous damping parameters. The common values that work well for a number of examples, using Real-ESSI, are:

```
1 a_0 = 10/s; // calibrated parameters
2 a_1 = 0.06*s; //calibrated parameters
3
4 soil_damping = 0.02; // change this value accordingly
5 add damping # 1 type Rayleigh with a0 = a_0*soil_damping a1 = a_1*soil_damping ←
    stiffness_to_use = Initial_Stiffness;
```
502.2.5 Free Field Ground Motions Development: Frequency Wave Number Method (Green’s functions) (fk)

Figure 502.12: 3D model.

Figure 502.13: 3D model, with observation point location.
Properties

- Model Properties
  - \(90 \text{ m} \times 90 \text{ m} \times 90 \text{ m}\) dimension
  - \(V_{s1} = 300 \text{ m/s}, V_{s2} = 400 \text{ m/s}\)
  - Poisson’s ratio1 = 0.25, Poisson’s ratio2 = 0.25
  - \(\text{Density1} = 940 \text{ kg/m}^3, \text{Density2} = 990 \text{ kg/m}^3\)

- Input Wave Properties
  - Generated using \(fk\) program
  - Variables are chosen to simulate Northridge Earthquake

Figure 502.14: Observation point displacements (top - EW, middle - NS, bottom UD) comparison between \(fk\) motions and DRM motions (that were developed using \(fk\) motions as input.

FEM Results, EW, NS, UD Components

Artificial 1D Downhole Array
Figure 502.15: Artificial 1D Downhole Array

Figure 502.16: EW Component, Station Depth = 0 m
Figure 502.17: Component, Station Depth = 50 m

Figure 502.18: Component, Station Depth = 0 m
Figure 502.19: Component, Station Depth = 50 m

Figure 502.20: Component, Station Depth = 0 m
Figure 502.21: Component, Station Depth = 50 m
502.2.6 Free Field Ground Motions Development: Fault Slip Model

![Fault Slip FE Model at -2km -2km.](image1)

Figure 502.22: Fault Slip FE Model at -2km -2km.

![Fault Slip FE Model at -2km -3km.](image2)

Figure 502.23: Fault Slip FE Model at -2km -3km.

![Fault Slip DRM Model.](image3)

Figure 502.24: Fault Slip DRM Model.

**Properties**
• Model Properties
  – $V_s = 700$ m/s
  – Poisson’s ratio1 = 0.1
  – Density = $1800$ kg/m$^3$

• Input Wave Properties
  – Generated using 2 different ‘Fault Slip Model’ (2 km $\times$ 2 km, 2 km $\times$ 3 km)
  – Ricker Wavelet (dominant frequency of 1 Hz)

Figure 502.25: Fault Slip Model, -2km -2km, (left) X and (right) Z displacements at middle top.

Figure 502.26: Fault Slip Model, -2km -3km, (left) X and (right) Z displacements at middle top.

Plane Wave Model
Figure 502.27: Fault Slip Model, -2km -2km, X displacements outside of Γ (DRM) domain.

Figure 502.28: Plane wave model.

Figure 502.29: 2D fault slip model disposition.

Figure 502.30: Seismic source mechanics. stress drop, Ormsby wavelet.
Figure 502.31: (left) Ormsby wavelet displacement and (right) Fourier transform. Used for stress drop.

Figure 502.32: Location of a measuring array, vertical, above source.

Figure 502.33: Accelerations along the measuring array above source: (left) horizontal, and (right) vertical.
Figure 502.34: Location of a measuring array, vertical, middle.

Figure 502.35: Accelerations along the measuring array in the middle: (left) horizontal, and (right) vertical.

Figure 502.36: Location of a measuring array, horizontal, surface.
Figure 502.37: Accelerations along the measuring array at the surface: (left) horizontal, and (right) vertical.
502.2.6.1 Input Motion

The point fault (element) is located at $X=3000\text{m}$ and $Z=3000\text{m}$ as shown in Fig. 502.113 (that is $2000\text{m}$ under and $2000\text{m}$ to the left of surface location of interest). The source is at the angle of $45^\text{deg}$, from the site at the middle top of the model. Pure shear is applied on the fault element in order to generate both S wave and P wave as a double couple source. In the actual analysis, equivalent nodal forces are used, and were obtained from constant surface tractions. Equivalent nodal forces for 27 nodes brick are shown in Fig. 502.38.

![Equivalent nodal forces](image)

Figure 502.38: Equivalent nodal forces to apply pure shear on 27 node brick element (Loading factors: $1/36$ for black, $4/36$ for blue, $16/36$ for red)

Ormsby wavelet is used as the time history of the shear force amplitude. The time history and its Fourier amplitude are shown in Figs. 502.39 and 502.40.
Figure 502.39: Time history of the Ormsby wavelet ($f_1 = 0$ [Hz], $f_2 = 1$ [Hz], $f_3 = 1$ [Hz], $f_4 = 1.5$ [Hz])

Figure 502.40: Fourier amplitude of the Ormsby wavelet ($f_1 = 0$ [Hz], $f_2 = 1$ [Hz], $f_3 = 1$ [Hz], $f_4 = 1.5$ [Hz])
502.2.6.2 Select Seismic Motions, Displacement Array Traces

The horizontal and vertical displacements, and particle motions in XZ plane at the site are observed in Fig. 502.41. Particle motions of every 1.2 sec are shown in the figure as well. Various color schemes were used for time section, to aid visualization of particle motions.

The point seismic source, as used in this case, a single finite element with cycles of pure shear, creates both primary (P, compressional) and secondary (S, shear). Initially only the P wave reaches the point of interest at the surface in the middle of the model, because of the radiation pattern and the site location (directly, at 45\(^{\circ}\)) away from the point source). On the other hand S wave propagates in the horizontal and vertical directions most strongly while there are traces of this wave away from these main propagation directions. With the P wave velocity \( V_p = 4899\text{m/s} \) (assumed of isotropic material) the first arrival time is 0.58 sec. Once the body waves hit the surface, surface (Rayleigh) wave is generated and propagates horizontally. Therefore, the obvious P wave particle motion can be seen only for the first 1.2 sec, while the ellipse orbit, which is the typical Rayleigh characteristic, is observed after 1.2 sec.
Figure 502.41: Displacement and particle motions at the site. (right top) Displacement time history, (left top) particle motion of the whole analysis, (bottom) particle motions of every 1.2 sec, S: Start, E: End.
To understand wave propagation better, displacement time history and particle motions are plotted on five observatory arrays.

- **Array1**: diagonal array from the fault to the site (Fig. 502.42)
- **Array2**: horizontal array from the source (Fig. 502.65)
- **Array3**: vertical array upon the fault (Fig. 502.77)
- **Array4**: horizontal array on the surface (Fig. 502.89)
- **Array5**: vertical array below the site (Fig. 502.101)

**Array1**: Displacement time history and particle motion on Array1 are shown in Fig. 502.43 to Fig. 502.53. Fig. 502.43 and Fig. 502.53 show the resultant motions at the site and the fault respectively. It is expected that the direct wave propagating along this array is only P wave, not S wave, because of the radiation pattern and the homogeneous medium. Nodes move in both horizontal and vertical direction together with the direct P wave. Dominant P wave can be seen at the beginning of the motion at every single nodes on this array. It is also observed that direct wave amplitude decreases when the observation point is far from the fault, a so called geometrical damping effect. Since direct S wave does not exist in this array, all non-diagonal displacements are caused by reflection wave and Rayleigh wave. Particle motion of Rayleigh wave is observed as ellipse in XZ plane. It is obviously confirmed that the effect of the Rayleigh wave is more significant as the observation point is near the surface. In order to make it easy to distinguish P and S wave propagation along the diagonal array, these resultant waves are converted into radial/transverse coordinate from vertical/horizontal coordinate (RT conversion). Converted waves of Array1 are shown in Fig. 502.54 to Fig. 502.64. It is now easy to observe P wave motion in radial direction and no S wave in Transverse direction.

**Array2 and Array3**: Results of Array2 are shown in Fig. 502.66 to Fig. 502.76 and results of Array3 are shown in Fig. 502.78 to Fig. 502.88. Since S wave is dominant in vertical and horizontal direction from the source while P wave is dominant in diagonal direction, only horizontal motion and vertical motion are observed well for the first several seconds until reflected wave and surface wave reach on these two arrays. Geometry damping effect is observed as well as P wave propagation in Array1. Rayleigh wave effects are propagating in depth. We can still see Rayleigh wave motion on these arrays, even though the effect of surface wave is less significant than Array1.
Array 4: Results of Array 4 are shown in Fig. 502.90 to Fig. 502.100. Array 4 is located on the surface and affected by surface wave effects the most.

Array 5: Results of Array 5 are shown in Fig. 502.102 to Fig. 502.112. It is sometimes assumed (by others) that waves propagate in vertical direction near the surface in case of horizontally layered ground model. In this case, the incident angle is 45 degree and the vertical wave propagation assumption is obviously not valid.

Figure 502.42: Observation array of Array 1
Figure 502.43: diagonal array X=5000m, Z=5000m

Figure 502.44: diagonal array X=4800m, Z=4800m
Figure 502.45: diagonal array X=4600m, Z=4600m

Figure 502.46: diagonal array X=4400m, Z=4400m
Figure 502.47: diagonal array X=4200m, Z=4200m

Figure 502.48: diagonal array X=4000m, Z=4000m
Figure 502.49: diagonal array X=3800m, Z=3800m

Figure 502.50: diagonal array X=3600m, Z=3600m
Figure 502.51: diagonal array X=3400m, Z=3400m

Figure 502.52: diagonal array X=3200m, Z=3200m
Figure 502.53: diagonal array $X=3000m$, $Z=3000m$
Figure 502.54: diagonal array X=5000m, Z=5000m

Figure 502.55: diagonal array X=4800m, Z=4800m
Figure 502.56: diagonal array X=4600m, Z=4600m

Figure 502.57: diagonal array X=4400m, Z=4400m
Figure 502.58: diagonal array X=4200m, Z=4200m

Figure 502.59: diagonal array X=4000m, Z=4000m
Figure 502.60: diagonal array X=3800m, Z=3800m

Figure 502.61: diagonal array X=3600m, Z=3600m
Figure 502.62: diagonal array X=3400m, Z=3400m

Figure 502.63: diagonal array X=3200m, Z=3200m
Figure 502.64: diagonal array X=3000m, Z=3000m
Figure 502.65: Observation array of Array2
Figure 502.66: X array2 X=5000m, Z=2950m

Figure 502.67: X array2 X=4800m, Z=2950m
Figure 502.68: X array2 X=4600m, Z=2950m

Figure 502.69: X array2 X=4400m, Z=2950m
Figure 502.70: X array2 X=4200m, Z=2950m

Figure 502.71: X array2 X=4000m, Z=2950m
Figure 502.72: X array2 X=3800m, Z=2950m

Figure 502.73: X array2 X=3600m, Z=2950m
Figure 502.74: X array2 X=3400m, Z=2950m

Figure 502.75: X array2 X=3200m, Z=2950m
Figure 502.76: X array2 X=3000m, Z=2950m
Figure 502.77: Observation array of Array3
Figure 502.78: Z array2 X=2950m, Z=5000m

Figure 502.79: Z array2 X=2950m, Z=4800m
Figure 502.80: Z array2 X=2950m, Z=4600m

Figure 502.81: Z array2 X=2950m, Z=4400m
Figure 502.82: Z array2 X=2950 m, Z=4200 m

Figure 502.83: Z array2 X=2950 m, Z=4000 m
Figure 502.84: Z array2 X=2950m, Z=3800m

Figure 502.85: Z array2 X=2950m, Z=3600m
Figure 502.86: Z array2 X=2950m, Z=3400m

Figure 502.87: Z array2 X=2950m, Z=3200m
Figure 502.88: Z array, X=2950m, Z=3000m
Figure 502.89: Observation array of Array4
Figure 502.90: X array 1 $X=5000m$, $Z=5000m$

Figure 502.91: X array 1 $X=4800m$, $Z=5000m$
Figure 502.92: X array1 X=4600m, Z=5000m

Figure 502.93: X array1 X=4400m, Z=5000m
Figure 502.94: X array1 X=4200m, Z=5000m

Figure 502.95: X array1 X=4000m, Z=5000m
Figure 502.96: X array1 X=3800m, Z=5000m

Figure 502.97: X array1 X=3600m, Z=5000m
Figure 502.98: X array1 X=3400m, Z=5000m

Figure 502.99: X array1 X=3200m, Z=5000m
Figure 502.100: X array1 X=3000m, Z=5000m
Figure 502.101: Observation array of Array5
Figure 502.102: Z array1 X=5000m, Z=5000m

Figure 502.103: Z array1 X=5000m, Z=4800m
Figure 502.104: Z array1 X=5000m, Z=4600m

Figure 502.105: Z array1 X=5000m, Z=4400m
Figure 502.106: Z array1 X=5000m, Z=4200m

Figure 502.107: Z array1 X=5000m, Z=4000m
Figure 502.108: Z array1 X=5000m, Z=3800m

Figure 502.109: Z array1 X=5000m, Z=3600m
Figure 502.110: Z array1 X=5000m, Z=3400m

Figure 502.111: Z array1 X=5000m, Z=3200m
Figure 502.112: Z array X=5000m, Z=3000m
502.2.6.3 Animations of Fault Slip Motions

All the motions are developed from a point source at a depth of −2km while the distance from the center of the model (where the observation point, or object of interest is) is 1km (27° off vertical), 2km (45° off vertical) and 3km (56° off vertical). There are three surface soil cases, homogeneous, single layer of soft soil and two layers of soft soil. Stress drop (input, pure shear at the one element) is defined by an Ormsby wavelet.

- Homogeneous soil/rock, 56° off vertical) (link to a movie, 39MB)
- Homogeneous soil/rock, 45° off vertical) (link to a movie, 32MB)
- Homogeneous soil/rock, 27° off vertical) (link to a movie, 37MB)
- Single layer soft soil with homogeneous soil/rock, 56° off vertical) (link to a movie, 30MB)
- Single layer soft soil with homogeneous soil/rock, 45° off vertical) (link to a movie, 34MB)
- Single layer soft soil with homogeneous soil/rock, 27° off vertical) (link to a movie, 32MB)
- Two layers of soft soil with homogeneous soil/rock, 56° off vertical) (link to a movie, 30MB)
- Two layers of soft soil with homogeneous soil/rock, 45° off vertical) (link to a movie, 31MB)
- Two layers of soft soil homogeneous soil/rock, 27° off vertical) (link to a movie, 32MB)

Details motions at the top of the model:

- Homogeneous soil/rock, 45° off vertical) motions at the top 2km × 2km (link to a movie, 30MB)
- Homogeneous soil/rock, 45° off vertical) motions at the very top, location of observation point and/or structure (link to a movie, 134MB)
502.2.6.4 Point Fault Slip Motions, Arrays and Particle Motions

The FEM model used is shown in Fig. 502.113. The brief description of FEM model is as follows:

- Model size: 10000m × 20m × 5000m
- Mesh size: 10m
- Element size: 20m (27 node brick element is used)
- Elastic parameters:
  - Poisson ratio $\nu = 0.4$
  - Shear wave velocity $V_s = 2000$ m/s
- Fixed boundary at $X = 0$ m, $X = 10000$ m, and $Z = 0$ m
- Free boundary at $Z = 5000$ m
- Plane strain condition in $y$-direction (all nodes on $y = 0$ m, $y = 20$ m are fully fixed).

Figure 502.113: FEM model (red square: fault element, blue circle: observation site)
502.3 Dynamic Soil-Foundation-Structure Interaction

Theoretical details for this section are given in Section 109.3 on page 535.

502.3.0.1 Animation of the DRM on a 1D Stack of Elements

(link to a movie, 8.9MB)
Using External Finite-Difference Seismic Code for DRM Motions

Introduction  This section explores the numerical conditions under which a high-performance fourth-order finite difference code for seismic modeling, henceforth the ‘seismic’ code, can be coupled successfully with a general purpose parallel non-linear finite element simulator, the FEM simulator, through the use of the domain reduction method (DRM). The approach taken consists in modeling a simple homogeneous half-space subjected to a single double-couple point-source to generate motions. DRM will be used to input these motions into the FEM simulator for an equivalent model of the domain, and the response compared at a control point on the surface. The seismic code used is SW4 (Sjögreen and Petersson, 2011) developed at Lawrence Livermore National labs, while the FEM simulator will be the Real-ESSI Simulator developed at the University of California Davis. Both are high-performance parallel programs highly regarded in their respective domains of application.

As originally proposed by Bielak et al. (2003a), the DRM input motions can be generated using a different method to compute the seismic wave field than the FEM code used to model site and structure. The rationale being that both methods will be approximating the same equations of elastodynamics and should both converge to the same solution as grid spacing tends to get smaller. What was implicit in that seminal work, but not explored or tested, is the effect of using different methods with possibly different orders of convergence, and how this affects convergence of the overall method.

The advantage of using different codes is that it is possible to choose a ‘seismic modeling’ code which is better suited and optimized for simulating earthquakes and then use the DRM to input the resulting motions into a code which is more suitable for modeling of non-linear soil and structural behavior in the chosen site. For example, SW4 has very convenient features for the input of double couple sources and also for extended sources, a task which would be much harder to achieve in a civil-engineering oriented code such as Real-ESSI Simulator. It is important to use the proper tools for modeling task, both for efficiency and credibility of the results used for design.

When solving the elastodynamic equations, it is expected that different solution schemes will yield different solutions for the same problem. Even if the seismic code’s finite difference grid points coincide spatially with the FE nodes, the different mathematical transformations involved in advancing the solution and the different orders of approximation will no-doubt lead to some degree of disagreement on the nodal values. When using the seismic code as DRM input into the FEM code it is expected that this disagreement will manifest itself in two ways: first, that the solution at a common control point within the DRM domain and on the seismic domain will differ and, second, consequently there will be a portion of the wave-field that will not be absorbed at the DRM boundary leading to outgoing motions which need to be damped out. Again, it is expected that both the difference in motions and the outgoing
wave-field will diminish with decreasing spatial and temporal discretization.

**Format for Node Coordinates, for Direct Extraction of Motions from SW4** This is based on Rodgers (2017).

A list of DRM nodes need to be supplied to SW4 modeler.

This list should be a simple 4-column asci file with node-name (6-10 characters), x, y, z location relative to origin (top, center) of ESSI domain.

**Example** Figure 502.115 illustrates the general layout of the DRM boundary elements with respect to the SW4 model for the domain used in this study. The seismic model consists on a single material elastic box of size $8 \text{ km} \times 4 \text{ km} \times 4 \text{ km}$, various values of the grid size are chosen ($h = 10 \text{ m}$ or $20 \text{ m}$), a point double-couple source is placed at $(2 \text{ km}, 2 \text{ km}, 2 \text{ km})$ such that it represents a reverse fault with a $45^\circ$ dip. The center of both models coincide. SW4 uses a right-handed coordinate system with $z$-axis positive downwards, while the FE meshes developed\(^1\) for Real-ESSI simulator use a right-handed system with $z$-axis upwards. Both models $x$-axis' and origins are made to coincide. Thus, the transformation from SW4 coordinates to Real-ESSI Simulator coordinates is:

$$
\begin{align*}
  x_{\text{SW4}} &= x_{\text{ESSI}} \\
  y_{\text{SW4}} &= -y_{\text{ESSI}} \\
  z_{\text{SW4}} &= -z_{\text{ESSI}}
\end{align*}
$$

Figure 502.115 shows the construction of the DRM model used in this study. It consists of an internal domain (within the DRM boundary) of dimensions $200 \text{ m} \times 200 \text{ m} \times 40 \text{ m}$ with discretization $dx = dy = dz = h = 5 \text{ m}$, $10 \text{ m}$ or $20 \text{ m}$. The elastic properties throughout both SW4 and Real-ESSI Simulator domains are such that the speed of P-waves is $V_p = 2000 \text{ m/s}$, the speed of S-waves is $V_s = 1000 \text{ m/s}$ and the density is $\rho = 2000 \text{ kg/m}^3$.

\(^1\)This is for convenience when using the meshing program *gmsh* to develop the mesh.
Table 502.1: Mesh information for DRM models used with Real-ESSI simulator.

<table>
<thead>
<tr>
<th>$h$, m</th>
<th>Number of Nodes</th>
<th>Number of elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.0</td>
<td>28,577</td>
<td>3,200</td>
</tr>
<tr>
<td>10.0</td>
<td>78,141</td>
<td>9,000</td>
</tr>
<tr>
<td>5.0</td>
<td>295,829</td>
<td>35,000</td>
</tr>
</tbody>
</table>

For the absorbent layer 4, 8 and 16 elements are used with Rayleigh damping ratios of $\xi = 0, 0.05, 0.1$. The Real-ESSI meshes are made up of second-order 27-node bricks with 27 Gauss-integration points. Table 502.1 summarizes the number of elements and nodes for the Real-ESSI Simulator meshes used.

The Real-ESSI meshes are generated using gmsh (Geuzaine and Remacle, 2009) and results are visualized using the custom visualizer plugin for ParaView (https://www.paraview.org/) (Ayachit, 2015).

Figure 502.116 shows the visualization of the displaced Real-ESSI mesh for $h = 20 \text{ m}$ (left) and $h = 10 \text{ m}$ (right) at time point $t = 3.40 \text{ s}$. No damping was used at the absorbing boundary at this point. This instant in time corresponds to the end of the arrival of the P-wave and, it can be observed, that DRM is working to eliminate the outgoing motions as the absorbing boundary has little or no displacement. Furthermore, the $h = 10 \text{ m}$ mesh seems to be doing a better job at absorbing the outgoing motions. Figure 502.117 shows the same situation for time point $t = 3.93 \text{ s}$. At this time the S-wave is coming into the domain. It can be seen that the $h = 20 \text{ m}$ domain is producing large outgoing motions while the finer domain is still handling the outgoing motions.

Figures 502.116 shows the trace plots for the control point at the center of the domain for both SW4 results as well as the results with DRM when using a DRM domain with discretization $h = 20 \text{ m}$ and $h = 10 \text{ m}$ respectively.

P-waves are faster than S-waves, in this case, by a factor of 2. Since wave resolvability for a given time-step size is proportional to wave speed, P-waves will be better resolved than S-waves. This manifests itself in the fact that initially, for P-wave arrivals, DRM works at capturing outgoing motions and later, for S-waves, some outgoing motions escape the boundary. These outgoing motions are not absorbed by damping since damping is not applied at this point, therefore, reflecting off the model boundaries back into the domain. This why there is oscillatory motion, corresponding to energy trapped in the system, observed after $t = 4 \text{ s}$. Note that the oscillatory motion is mainly seen in the $x$-component and not the others.

To mitigate these trapped waves, the absorbent boundary is assigned Rayleigh damping of varying intensities. Rayleigh damping can be applied in many ways. From (Tafazzoli, 2012) several lessons can
be drawn regarding how to design the absorbent boundary for maximum efficiency. The following is a summary of these lessons:

- Increasing in-absorbent-boundary damping has the general effect of reducing the amplitude of waves reflected at the domain boundary back into the internal domain.

- Sharp damping contrasts produce additional reflected waves at the DRM/absorbing boundary interface.

- Gradually increasing the damping ratio with increasing distance to the DRM boundary alleviates the reflection issue while retaining a similar damping efficiency.

- Increasing the thickness in elements of the absorbent boundary also increases the damping efficiency.

- When selecting two frequencies to provide for specification of Rayleigh damping, the best frequency is not related to the ‘natural’ frequency of the soil stratum as is commonly assumed in practice and some research. The best frequency is related, instead, to the frequency of the out-going waves.

In this study a uniform value for the damping ratio assigned to the Rayleigh damping coefficients is used. Figure 502.119 shows the effect that this additional damping has on DRM models with different sizes. The beneficial effect of this damping in capturing the energy leaking out from the DRM boundary is apparent.

Perfect matching of the motions obtained with SW4 and Real-ESSI simulation with DRM modeling was not achieved. A key reason for this is that the SW4 simulations were done at $h = 20$ m while varying the mesh size for the Real-ESSI simulations. This means that SW4 motions had to be interpolated between grid spaces when the grids did not match. SW4 provides only ‘nearest’ neighbor interpolation for requested output stations, so an improvement on this is needed if better matching is expected. Alternatively, it is possible to achieve better agreement by decreasing the discretization on both domains.

From these experiments it is possible to extract the following design considerations when seeking to couple DRM-based finite element simulations with other forms of seismic modeling:

- Size of the DRM domain is irrelevant for the ‘free-field’ problem.

- Relative order of accuracy of the Finite-element mesh and the code that produces seismic input is important when designing mesh sizes for both the finite-element and the seismic simulation.

- Matching motions perfectly might result in expensive computations, beyond what is needed due to physical and numerical constraints for the propagation problem alone.
• Out-going motions due to mismatch need to be absorbed outside the DRM domain by some method.

Ultimately, the purpose of using the DRM is as the enabling technology allowing the rational modeling of perturbations of the free-field model in order to reduce the cost of jointly modeling complex site and structure response along with the seismic wave propagation problems. In such a case, out-going motions will be unavoidable and have to be dealt with efficiently. In any such study it will be very important to demonstrate that the DRM motions and forces developed agree with the free-field model as a basic way of showing the adequacy of the numerical implementations involved in the modeling effort.

**Practical considerations** Using SW4 to generate motions for DRM in Real-ESSI requires the following steps:

1. Generate a large-scale geologic model in SW4.
2. Generate a FEM mesh for Real-ESSI, generate the following lists:
   - Coordinates and numbers of nodes in DRM layer.
   - A flag indicating whether a particular node is internal or external.
   - List of elements in DRM layer.
3. Within the SW4 input file write recorder lines (USGS format) for all DRM nodes. Example:
   ```
   rec x=4037.5 y=2050.0 z=40.0 file=node000734b sacformat=0 usgsformat=1
   ```
4. Run SW4.
5. Collect the results for each recorded node in one DRM HDF5 input file.

Some notes to complement the above steps.

- Nodes in SW4 input file must have coordinates in SW4 coordinates. Remember that, in SW4, down is positive. The best is to develop FEM model centered around origin of a coordinate system, so that determining the position in SW4 coordinates is a simple translation and flipping of the signs for both Z and Y coordinates.
- Use the SW4 output file name in a meaningful way to determine which node number it belongs to. In the example nodeXXXe where XXX has the node number and e is either e or b depending if it is an internal (boundary) node or external.
• Remember to reverse the sign of Y and Z results when writing the DRM HDF5 input file from the SW4 results.

• It is recommended to use chunking and compression in HDF5 dataset to speed up loading times and optimize storage usage.

Template code for generating HDF5 input suitable for DRM analysis in Real-ESSI.

```python
import h5py
import scipy as sp
import time

#Global parameters
Ntimesteps = 1000
dt = 0.0001
Nnodes = 10000
Nelements = 4000

#Create HDF5 file - note filename
h5file = h5py.File("earthquake.h5.drminput","w")

#Initialize memory
u = sp.zeros((3*Nnodes, Ntimesteps), dtype = sp.double) #Will hold dispalcements
a = sp.zeros((3*Nnodes, Ntimesteps), dtype = sp.double) #Will hold accelerations
nodelist = sp.zeros(Nnodes,dtype=sp.int32)
is_boundary_node = sp.zeros(Nnodes,dtype=sp.bool_)
elements = sp.zeros(Nelements,dtype=sp.int32)

#Time vector
t = sp.linspace(0,dt*(Ntimesteps-1),dt)

# =========================
# =========================
# * Read in u from SW4 output, determine v and a (differentiate).
# * Remember to flip signs for Y and Z components.
# * Also read in:
# - is_boundary_node (boolean vector, see above)
# - nodelist (integer vector, see above)
# - elements (integer vector, see above)
# This is problem and formatting specific and likely to change depending on many factors.
```
# Count numbers of nodes (DRM input file needs this)

\[\text{Nb} = 0\]
\[\text{Ne} = 0\]

for i in range(Nnodes):
    if is_boundary_node[i] == True:
        Nb += 1
    else:
        Ne += 1

# Write out HDF5 file.

h_acc = h5file.create_dataset("Accelerations", (3*Nnodes,Ntimesteps),
                               dtype=sp.double, data=a)

h_dis = h5file.create_dataset("Displacements", (3*Nnodes,Ntimesteps),
                               dtype=sp.double, data=u)

h5file.create_dataset("Time", data=t)

h5file.create_dataset("Elements", data=elements)

h5file.create_dataset("DRM Nodes", data=nodelist)

h5file.create_dataset("Is Boundary Node", data=is_boundary_node,
                      dtype=sp.int32)  # This array has 1 if the node at the corresponding position
                      # in "DRM nodes" array is a boundary node and zero if not

h5file.create_dataset("Number of Exterior Nodes", data=Ne)

h5file.create_dataset("Number of Boundary Nodes", data=Nb)

# For big cases, it is better to do this one record at a time.

# Write timestamp (time format used is that of c "asctime" Www Mmm dd hh:mm:ss ←
# yyyy example: Tue Jan 13 10:17:09 2009)

localtime = time.asctime( time.localtime(time.time()) )

h5file.create_dataset("Created", data=str(localtime))

# Close HDF5

h5file.close()
Figure 502.115: Free field DRM model for verification of SW4 and Real-ESSI Simulator coupling. From top, internal domain, DRM boundary, absorbent element layer, complete model.
Figure 502.116: Visualization of the DRM solution at time $t = 3.40\,s$ for DRM mesh sizes of $h = 20\,m$ (left) $10\,m$ (right). SW4 mesh is $h = 20\,m$ for both cases. Color shows magnitude of displacement vector. Arrival of P wave is correctly resolved on both meshes.

Figure 502.117: Visualization of the DRM solution at time $t = 3.93\,s$ for same setup as Figure 502.116, with color showing magnitude of the displacement vector. Arrival of S wave is better resolved on the $h = 10\,m$ mesh but not on the $h = 20\,m$ mesh as can be seen by looking at the out-going motions. SW4 mesh is $h = 20\,m$ for both cases.
Figure 502.118: Generated motions at the center node \((x = 4000\,m, y = 2000, z = 0)\), blue blue shows SW4 motions generated with an \(h = 2\,m\) grid, and my orange orange shows motions obtained with Real-ESSI using DRM and variable mesh size.
Figure 502.119: Results from computing motions with Real-ESSI Simulator within the DRM domain for different ESSI mesh sizes $h$ and Rayleigh damping with $\xi = 0.1$. All SW4 motions were computed for $h = 20$ m.
502.3.1 Seismic Wave Propagation Modeling and Simulation: Numerical Accuracy and Stability

The accuracy of a numerical simulation of dynamic SFSI is controlled by two main parameters: a) the spacing of the nodes of the finite element model ($\Delta h$) and b) the length of the time step $\Delta t$. Assuming that the numerical method converges toward the exact solution as $\Delta t$ and $\Delta h$ go toward zero the desired accuracy of the solution can be obtained as long as sufficient computational resources are available.

502.3.1.1 Grid Spacing $\Delta h$

In order to represent a traveling wave of a given frequency accurately about 10 nodes per wavelength $\lambda$ are required for finite element with linear displacement interpolations between nodes\(^2\) (Bathe and Wilson, 1976; Hughes, 1987; Argyris and Mlejnek, 1991). Fewer than 10 nodes can lead to numerical damping as the discretization misses certain peaks of the wave. In order to determine the appropriate maximum grid spacing the highest relevant frequency $f_{\text{max}}$ that is present in the model needs to be found by performing a Fourier analysis of the input motion. Typically, for seismic analysis $f_{\text{max}}$ is about 10 Hz. By choosing the wavelength $\lambda_{\text{min}} = v/f_{\text{max}}$, where $v$ is the wave velocity, to be represented by 10 nodes the smallest wavelength that can still be captured partially is $\lambda = 2\Delta h$, corresponding to a frequency of $5 f_{\text{max}}$. This is true for finite elements with linear interpolation of displacements.

In general, for finite elements with different interpolation of displacements the following conclusion can be made about the number of elements used for proper wave propagation:

- Linear interpolation finite elements (1D 2-node truss, 2D 4-node quad, 3D 8-node brick): $h^{LE} \leq v/(10 f_{\text{max}})$
- Quadratic interpolation finite elements (1D 3-node truss, 2D 9-node quad, 3D 27-node brick): $h^{QE} \leq v/(2 f_{\text{max}})$
- Structural elements (beams, shells) have at least quadratic interpolation functions (Euler-Bernoulli beam uses cubic Hermite polynomials) and since material is very stiff, and wave propagation speed is very high, these elements usually satisfy wave propagation criteria a priori.

For example, element size for propagating $f_{\text{max}} = 20$ Hz, for linear interpolation element $\Delta h^{LE} \leq v/200$ Hz while for quadratic interpolation element $\Delta h^{QE} \leq v/40$ Hz. When material plastifies, element size needs reduction, depending on the reduction in (shear) wave velocity.

\(^2\) If quadratic finite elements are use, for example a 27 node brick than only two elements are needed per wave length as they can properly represent the single wave (as they have quadratic interpolation for displacements, and feature 5 along the wave).
502.3.1.2 Time Step Length $\Delta t$

The time step $\Delta t$ used for numerically solving nonlinear vibration or wave propagation problems has to be limited for two reasons. The stability requirement depends on the numerical procedure in use and is usually formulated in the form $\Delta t/T_n < \text{value}$. $T_n$ denotes the smallest fundamental period of the system. Similar to the spatial discretization $T_n$ needs to be represented by about 10 time steps. While the accuracy requirement provides a measure on which higher modes of vibration are represented with sufficient accuracy, the stability criterion needs to be satisfied for all modes. If the stability criterion is not satisfied for all modes of vibration, then the solution may diverge. In many cases it is necessary to provide an upper bound to the frequencies that are present in a system by including frequency dependent damping to the model.

The second stability criterion results from the nature of the finite element method. As a wave front progresses in space it reaches one point after the other. If the time step in the finite element analysis is too large the wave front can reach two consecutive elements at the same moment. This would violate a fundamental property of wave propagation and can lead to instability. The time step therefore needs to be limited to

$$\Delta t < \frac{\Delta h}{v}$$

(502.2)

where $v$ is the highest wave velocity.

502.3.1.3 Nonlinear Material Models

If nonlinear material models are used the considerations for stability and accuracy as stated above don't necessarily remain valid. Especially modal considerations need to be examined further for these cases. It is however save to assume that the natural frequencies decrease as plastic deformations occur. The minimum time step required to represent the natural frequencies of the dynamic system can therefore taken to be the same as in an elastic analysis.

A high frequency component is introduced due to plastic slip and counter balancing of the resulting displacement. This is especially true if a linear algorithm with no iterations within one time step is used. Figure 502.120 shows a part of an acceleration time history from an analysis involving elastic-plastic material. It can be seen that the out-of-balance forces at the end of a time step can be quite large if a linear algorithm is used. While the Newton-Raphson algorithm minimizes out-of-balance forces within one time step the linear algorithm requires several time steps to return to a stable equilibrium path.

The frequencies corresponding to these peaks are typically of the order of $1/(\text{a few } \Delta t)$. Normally the time step is small enough so that these frequencies don’t interfere with the input motion. They can
be prevented from propagating through the model by an appropriate choice of algorithmic or material damping.

For stability the time step used in a nonlinear analysis needs to be smaller than in a linear elastic analysis. By how much it has to be reduced is difficult to predict as this depends on many factors such as the material model, the applied loading or the numerical method itself. Argyris and Mlejnek (1991) suggest the time step to be reduced by 60% or more compared to the time step used in an elastic analysis. The best way to determine whether the time step is appropriate for a given analysis consists in running a second analysis with a reduced time step.

502.3.2 Seismic Wave Propagation Modeling and Simulation: Domain Boundaries

One of the biggest problems in dynamic SFSI in infinite media is related to the modeling of domain boundaries. Because of limited computational resources the computational domain needs to be kept small enough so that it can be analyzed in a reasonable amount of time. By limiting the domain however an artificial boundary is introduced. As an accurate representation of the soil-structure system this boundary has to absorb all outgoing waves and reflect no waves back into the computational domain. The most commonly used types of domain boundaries are presented in the following:

- Fixed or free

By fixing all degrees of freedom on the domain boundaries any radiation of energy away from the structure is made impossible. Waves are fully reflected and resonance frequencies can appear that don’t exist in reality. The same happens if the degrees of freedom on a boundary are left ‘free’, as at the surface of the soil.
A combination of free and fully fixed boundaries should be chosen only if the entire model is large enough and if material damping of the soil prevents reflected waves to propagate back to the structure.

- Absorbing Lysmer Boundaries

A way to eliminate waves propagating outward from the structure is to use Lysmer boundaries. This method is relatively easy to implement in a finite element code as it consists of simply connecting dash pots to all degrees of freedom of the boundary nodes and fixing them on the other end (Figure 502.121).

![Figure 502.121: Absorbing boundary consisting of dash pots connected to each degree of freedom of a boundary node](image)

Lysmer boundaries are derived for an elastic wave propagation problem in a one-dimensional semi-infinite bar. It can be shown that in this case a dash pot specified appropriately has the same dynamic properties as the bar extending to infinity (Wolf, 1988). The damping coefficient $C$ of the dash pot equals

$$C = A \rho c$$  \hspace{1cm} (502.3)

where $A$ is the section of the bar, $\rho$ is the mass density and $c$ the wave velocity that has to be selected according to the type of wave that has to be absorbed (shear wave velocity $c_s$ or compressional wave velocity $c_p$).

In a 3d or 2d model the angle of incidence of a wave reaching a boundary can vary from almost $0^\circ$ up to nearly $180^\circ$. The Lysmer boundary is able to absorb completely only those under an angle of incidence of $90^\circ$. Even with this type of absorbing boundary a large number of reflected waves...
are still present in the domain. By increasing the size of the computational domain the angles of incidence on the boundary can be brought closer to $90^\circ$ and the amount of energy reflected can be reduced.

- Infinite elements

- More sophisticated boundaries modeling wave propagation toward infinity (boundary elements)

For a spherical cavity involving only waves propagating in radial direction a closed form solution for radiation toward infinity, analogous to the Lysmer boundary for wave propagation in a prismatic rod, exists (Sections 3.1.2 and 3.1.3 in Wolf (1988)). Since this solution, in contrast to the Lysmer boundary, includes radiation damping it can be thought of as an efficient way of eliminating reflections on a semi-spherical boundary surrounding the computational domain.

More generality in terms of absorption properties and geometry of the boundary are provided by the various boundary element methods (BEM) available in the literature.

502.3.3 Soil/Rock Modeling and Simulation

502.3.4 Soil/Rock – Foundation Contact (Slipping and Gaping) Modeling and Simulation

502.3.5 Buoyancy Modeling and Simulation

For self weight and other static loads, and if we assume infinitely stiff structures, buoyant force $B$ can be calculated and applied as a single resultant force directed upward. Calculation of static buoyant force is based on the Archimedes principle: "Any object immersed in water is buoyed up by a force equal to
the weight of water displaced by the object", and therefore such a buoyancy force is given as:

\[ B = \rho_w g V \]  \hspace{1cm} (502.4)

where \( \rho_w = 999.972 \text{ kg/m}^3 \) is the mass density of water (at temperature of +4°C with small changes of less than 1% up to +40°C). \( g = 9.81 \text{ m/s}^2 \) is the gravitational acceleration, and \( V \) is the volume of displaced fluid.

For seismic and other dynamic loading, and for structures which have finite stiffness (real stiffness), effects of buoyant (pressure) force will have to be modeled and simulated using methods that are more sophisticated than the simple static approach noted above.

Two main approaches to buoyancy modeling are described below:

- **Sharp contrast in permeability (naturally occurring)**, between soil/rock and foundation concrete can be used to model buoyant pressures (and consequentially buoyant forces). This approach has been used before (Cheng and Jeremić, 2009b) and it works quite satisfactory if gap is not expected to form between soil/rock and concrete. Using sharp contrast approach, the buoyant forces are created by providing physical permeabilities for soil/rock (permeable rock, which can have high permeability) and for concrete which is quite impermeable (but not absolutely impermeable, permeability of concrete is couple of orders of magnitude lower than that of soil/rock). When the water tries to move (natural process due to pore water pressure gradient), it is restricted by the low permeability of the concrete, and thus forms a region of pressure (hydrostatic for static loading, or a different, dynamic pressure that results from dynamic behavior of soil/rock). This pressure is actually acting as a buoyant force on the concrete foundation. However, this approach only works well when there is no gap opening. In addition to that, there are modeling problems, with high pressure gradient close to the boundary between soil/rock and concrete, modeler needs to carefully mesh that region, to overcome too large pressure gradients in single layer of finite elements. A better approach, with or without gap opening is to use special coupled contact/interface finite element described below.

- **Special coupled contact/interface finite element explicitly models water displacements and pressures and allows for explicit gap opening, filling of gap with water, slipping (frictional) when the gap is closed, and pumping of water as gap opens and closes.** This contact/interface element incorporates the pore water pressure information, as well as the information about the displacement (movement) of pore water within a gap. It is based on a previous version of the contact/interface element with two important features in addition to features available to the dry contact element:

  - Pore water pressure values (physical values obtained from simulation) from one side of the
special coupled contact/interface element (side in contact/interface with saturated soil/rock) will be communicated (directly transferred) to the other side of this element. With this water pressure information available, the applied water pressure will be acting on the foundation finite elements. Foundation finite element will integrate pressure field on a given face (or faces) of the element and will create a buoyant force. Integration of pressure on an element face into buoyant forces is done using standard finite element procedure for calculating nodal forces (these are the buoyant forces in our case) from face pressures (Jeremić et al., 1989-2021):

$$F_{Ia}^{buoyant} = \int_{S_m} f_{Ia}^{buoyant} H_I \, dS_m$$

(502.5)

where $F_{Ia}^{buoyant}$ is the buoyant force at each node of the foundation finite element, $f_{Ia}^{buoyant}$ is the distribution of buoyant pressures on a face the foundation finite element, and $H_I$ is a standard shape function of the foundation finite element (in our case, linear for 8 node brick, or quadratic for a 27 node brick). The integration is performed over a surface area $S_m$ of each finite element face where buoyant pressures are present. With buoyant forces acting at the bottom (or sides) of a foundation, a proper reduction (change) of contact/interface pressures (forces) will be calculated. This means that the slipping (frictional) criteria of the contact/interface element will have all the necessary information about the normal forces (now reduced because of buoyant forces) and will determine is the contact/interface will slip and remain attached (at the location of that contact/interface element).

- Gap opening (physical values obtained from simulations) will be used to create suction and compression pore/gap water field. This will be achieved by connecting the displacements of the pore fluid from soil/rock finite elements (we will be using u-p-U finite elements for soil/rock modeling (Jeremić et al., 2008)), to the contact/interface element node on the opposite side of the special coupled contact element. This way, if the gap opens, and the contact/interface element now features an opening, the water displacements from the soil/rock side of the contact/interface element will be "pulled" to follow the uplifting foundation. This water movement will create pressure gradients in the soil/rock elements beneath, which will be accurately modeled using fully coupled u-p-U finite elements for soil/rock (Jeremić et al., 2008; Jeremić and Cheng, 2009).

Using above described approach to modeling will provide for high fidelity modeling and simulation of the buoyant pressures/forces, which will resolve all the difficulties related to this modeling.
502.3.6 Structural Foundations Modeling and Simulation

27 node solid bricks
502.3.7 Seismic Isolator Modeling and Simulation

Latex Rubber
- Neoprene Rubber
- Rubber with lead core
- Frictional Pendulum
502.3.8 Structural Components Modeling and Simulation

Shells

Thick shells
502.3.9 Nonlinear Time Domain Analysis Progress and Example

502.3.9.1 Model Development

Mesh Development

Material Model Development

Loading Stages Development

502.3.9.2 Simulation Development

Sequential versus Parallel

Simulation Progress and Control

502.3.9.3 Seismic Motions

Full 3C Seismic Wave Field

- 1C wave field (deconvolution)
- 3C, inclined or vertical (body and surface waves) wave field, using 2D FEM fault slip model, see section 502.2.6 on page 2217
- 3C, inclined (body and surface waves) wave field, using fk (see section 706.1 on page 2759
- 3C, inclined (body and surface waves) wave field, using analytic solutions (Kausel, 2006)

Seismic Input Using DRM Theory, see section 109.4.1.1 on page 538.

Input, see section section 205.3 on page 791

502.4 Step by Step, Hierarchical Inelastic ESSI Modeling and Simulation

Recommended nonlinear/inelastic modeling and simulation phases for an SSI system are shown in Figure 502.122.

It is highly recommended to proceed in phases, from simpler to more sophisticated, while employing sound engineering judgement at the end of each phase in order to understand static and dynamic response
of the system and its components. For example, for an SSI system shown in Figure 502.122, advice is to follow these steps:

- start with a 1D model for one component (1C) shear wave, SV free field wave propagation, linear elastic soil, simple wavelet motions (Ricker, Ormsby)
- For a 1D-1C free field wave propagation, linear elastic soil, use more realistic motions, using deconvolution of surface motions and convolution of rock motions from depth
• For a 1D model, apply 1 component (1C) vertical motions, a free field P wave

• For a 1D model, apply 2 component (2C), shear wave, SV, motions, for a free field wave propagation,

• For a 1D model, apply 3 component (3×1C) motions, for a free field wave

• For a 1D-1C, 1D-2C and 1D-3×1C free field wave propagation, update soil model to mild nonlinear/inelastic and perform wavelet and seismic motions, as noted above

• For a 1D-1C, 1D-2C and 3×1C models, update soil model to more realistic nonlinear/inelastic model and perform wavelet and seismic motions, as noted above

• Develop a 3D model of a free field and follow above steps, 3D-1C, 3D-2C, 3D-3×1C, elastic, inelastic, wavelet, realistic seismic input

• For a 3D model develop full 3C motions, inclined waves, and test your system, using phased modeling of material (elastic-inelastic), and motions, wavelets, then realistic seismic motions

• For a 3D-1C, 3D-2C and 3D-3×1C, models, add foundation slab, follow steps from above, elastic, inelastic soil, wavelets, realistic motions.

• Develop model of a structure, fully fixed DoFs at the bottom and perform eigen-analysis.

• For a fixed base structural model, apply motions from above,

• Progression in modeling sophistication from simpler material models, linear elastic, to more sophisticated, inelastic material models, should be followed for each component of the model, soils, structures, special elements, &c.

• Finally you might be able to develop a full 3D model, first linear then slowly nonlinear/inelastic, first simple motions, wavelets, then more realistic 1C, 2C, 3×1C and 3C motions

Progression in modeling sophistication from simpler material models, linear elastic, to more sophisticated, inelastic material models, as suggested above, should be followed by progression of simulation sophistication, as noted below:

• Start with linear elastic material models for all components, including bonded contact/interface/joints.

• Proceed with explicit, non convergence check simulation at simulation at both constitutive and global levels:
On constitutive level, use Forward Euler algorithm. Theory for the Forward Euler, explicit constitutive algorithm is described in Section 104.3, on page 192, in Jeremić et al. (1989-2021). Commands for constitutive level Forward Euler algorithm are given in Section 205.3.5.15, on page 1075, in Jeremić et al. (1989-2021).

With Forward Euler, explicit computations on the constitutive level, it is only appropriate to use global, finite element algorithm with no convergence check, that is also known as the explicit global algorithm. This algorithm is described in some detail in Section 107.3, on page 492, in Jeremić et al. (1989-2021). Commands for constitutive level Forward Euler algorithm are given in Section 205.3.5.14, on page 1074, in Jeremić et al. (1989-2021).

For fully implicit algorithm, with enforcement of equilibrium on constitutive and global, finite element levels, proceed with Backward Euler (or one of the variants) on constitutive level and Newton algorithm (or one of variants) on global level:

On constitutive level, use Backward Euler algorithm. Theory for the Backward Euler, implicit constitutive algorithm is described in Section 104.4, on page 193, in Jeremić et al. (1989-2021). Commands for constitutive level Forward Euler algorithm are given in Section 205.3.5.15, on page 1075, in Jeremić et al. (1989-2021).

With Backward Euler, implicit computations on the constitutive level, it is appropriate to use global, finite element algorithm with or without convergence check, that is, one can use either explicit or implicit global algorithm. These algorithm is described in some detail in Section 107.3, on page 492, in Jeremić et al. (1989-2021). Commands for constitutive level Backward Euler algorithm are given in Section 205.3.5.14, on page 1074, in Jeremić et al. (1989-2021).

A section from IAEA TECDOC chapter will be used for this section ...

Pecker et al. (2021)

Input files for all the examples are available online at this LINK. All the examples can run directly at the Amazon Web Services, through Real-ESSI image.
Chapter 503

Earthquake-Soil-Structure Interaction, Bridge Structures

(2003-2007-2011-)

(In collaboration with Dr. Guanzhou Jie)
503.1 Chapter Summary and Highlights
503.2 Case History: Earthquake-Soil-Structure Interaction for a Bridge System

503.2.1 Prototype Bridge Model Simulation

The final objective of this work is to improve current modeling techniques through the comparative study between numerical and experimental components. In this work, a whole prototype bridge model has been built using finite element techniques demonstrated in previous sections. Domain Reduction Method (Bielak et al., 2003b; Yoshimura et al., 2003b) (DRM) has been used to reduce the model size while still preserving the accuracy of the ground motion analysis.

503.2.1.1 Soil Model

Capitol Aggregates, a local quarry located in the south of Austin, has been selected to be the test site for this project. Site characterization has been performed to collect information on the soil (Kurtulus et al., 2005). Based on the only triaxial test data available, a nonlinear soil constitutive model is developed in this work for prototype finite element analysis.

Undrained triaxial compression test has been carried out on one, 1.5 in diameter triaxial test specimen trimmed from an undisturbed soil sample obtained from borehole at an approximate depth of 10.6 ft. The initial size and index properties of the soil specimen are given in Table 503.1 (Kurtulus et al., 2005).

<table>
<thead>
<tr>
<th>Soil Index Property</th>
<th>Initial</th>
<th>After Consolidation</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter D (inch)</td>
<td>1.50</td>
<td>1.48</td>
<td>1.56</td>
</tr>
<tr>
<td>Height H (inch)</td>
<td>3.00</td>
<td>2.87</td>
<td>2.56</td>
</tr>
<tr>
<td>Total Unit Weight $\gamma_t$ (pcf)</td>
<td>107.3</td>
<td>111.1</td>
<td>112.8</td>
</tr>
<tr>
<td>Water Content $w$ (%)</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Dry Unit Weight $\gamma_d$ (pcf)</td>
<td>90.9</td>
<td>94.3</td>
<td>95.7</td>
</tr>
<tr>
<td>Void Ratio $e^1$</td>
<td>0.84</td>
<td>0.77</td>
<td>0.75</td>
</tr>
<tr>
<td>Degree of Saturation $S_r^1$ (%)</td>
<td>57</td>
<td>62</td>
<td>64</td>
</tr>
</tbody>
</table>

$^1$Specific Gravity $G_s$ is assumed to be 2.68.

In the triaxial cell, the specimen was allowed to come into equilibrium (compress/consolidate with drainage lines open) under an isotropic pressure equal to the assumed in-situ mean total stress, which is about 5.6 psi. Upon equilibrating, the specimen was sheared under undrained conditions with a strain rate of $\%1$ per hour. No pore pressure readings were taken since the specimen was unsaturated. The
resulting stress-strain curve is presented in Figure 503.1. An estimate of the undrained shear strength in terms of total stresses was measured as 13.41 psi (1931 psf) at about 9% strain. The specimen failed in a bulging mode. The index properties of the specimen at failure are presented in Table 503.1.

![Figure 503.1: Total Stress Strain Curve Determined from Undrained Triaxial Compression Test (Undisturbed Sample from Depth 10.6 ft)](image)

Based on the laboratory triaxial test data, a nonlinear elastic-plastic soil model has been developed to calibrate the finite element simulation. Associative Drucker-Prager plasticity model, combined with nonlinear Armstrong-Frederick kinematic hardening rule, yields good match between laboratory data and numerical results, as shown in Figure 503.2. The same model has been exposed to various confinements to test robustness of the model for soils at different depths.

503.2.1.2 Element Size Determination

The accuracy of a numerical simulation of dynamic SFSI (Soil-Structure-Foundation-Interaction) problems is controlled by two main parameters (Preisig, 2005):
Figure 503.2: Total stress strain curve obtained using simulation of triaxial test (from depth 10.6 ft)

1. The spacing of the nodes of the finite element model $\Delta h$

2. The length of the time step $\Delta t$.

Assuming that the numerical method converges toward the exact solution as $\Delta t$ and $\Delta h$ go toward zero the desired accuracy of the solution can be obtained as long as sufficient computational resources are available.

As presented in Preisig (2005), in order to represent a traveling wave of a given frequency accurately about 10 nodes per wavelength are required. Fewer than 10 nodes can lead to numerical damping as the discretization misses certain peaks of the wave. In order to determine the appropriate maximum grid spacing the highest relevant frequency $f_{\text{max}}$ that is present in the model needs to be found by performing a Fourier analysis of the input motion. Typically, for seismic analyses $f_{\text{max}}$ is about 10 Hz. By choosing the wavelength $\lambda_{\text{min}} = v/f_{\text{max}}$, where $v$ is the wave velocity, to be represented by 10 nodes the smallest wavelength that can still be captured partially is $\lambda = 2\Delta h$, corresponding to a frequency of $5f_{\text{max}}$.

The maximum grid spacing should not exceed

$$\Delta h \leq \frac{\lambda_{\text{min}}}{10} = \frac{v}{10f_{\text{max}}}$$

(503.1)
Table 503.2: Maximum Element Size Determination ($f_{max} = 10Hz$)

<table>
<thead>
<tr>
<th>Depth (ft)</th>
<th>Thickness (ft)</th>
<th>$v_{shear}$ (fps)</th>
<th>$\Delta h_{max}$ (ft)</th>
<th>$h_{max}$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>320</td>
<td>3.2</td>
<td>0.98</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>420</td>
<td>4.2</td>
<td>1.28</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
<td>540</td>
<td>5.4</td>
<td>1.65</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>660</td>
<td>6.6</td>
<td>2.01</td>
</tr>
<tr>
<td>14</td>
<td>7.5</td>
<td>700</td>
<td>7.0</td>
<td>2.13</td>
</tr>
<tr>
<td>21.5</td>
<td>17</td>
<td>750</td>
<td>7.5</td>
<td>2.29</td>
</tr>
<tr>
<td>38.5</td>
<td>half-space</td>
<td>2200</td>
<td>22.0</td>
<td>6.7</td>
</tr>
</tbody>
</table>

where $v$ is the smallest wave velocity that is of interest in the simulation.

In this work, the prototype site chosen is Capitol Aggregates, a local quarry located in the south of Austin. According to the site characterization report Kurtulus et al. (2005), we obtain Table 503.2 for element size determination.

Mechanical properties of soil changes with cyclic loadings. In order to predict more accurately the dynamic behaviors of soil subject to earthquake loadings, various laboratory and in situ tests have been performed to examine the degradation of dynamic soil properties. Equivalent linear model has been used extensively in practice (Kramer, 1996b). Moduli reduction curve ($G/G_{max}$) and damping ratio relationship have been obtained for prototype soil at the site of Capitol Aggregates as shown in Figure 503.3.
Figure 503.3: Comparison of the Variation in Normalized Shear Modulus and Damping Ratio with Shearing Strain from the Resonant Column Tests with Modulus Reduction Curves proposed by Seed et al. (1986) and Darendeli (2001) (Kurtulus et al., 2005)
Table 503.3: Element Size Determination after Degradation of $G(v_s) \ (f_{\text{max}} = 10Hz)$

<table>
<thead>
<tr>
<th>Depth (ft)</th>
<th>Thickness (ft)</th>
<th>$v_s$ (fps)</th>
<th>Min $G/G_{\text{max}}$</th>
<th>Min $v_s$</th>
<th>$\Delta h_{\text{max}}$ (ft)</th>
<th>$h_{\text{max}}$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>320</td>
<td>0.36</td>
<td>192</td>
<td>1.92</td>
<td>0.59</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>420</td>
<td>0.36</td>
<td>252</td>
<td>2.52</td>
<td>0.77</td>
</tr>
<tr>
<td>2.5</td>
<td>4.5</td>
<td>540</td>
<td>0.36</td>
<td>324</td>
<td>3.24</td>
<td>0.99</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>660</td>
<td>0.36</td>
<td>396</td>
<td>3.96</td>
<td>1.21</td>
</tr>
<tr>
<td>14</td>
<td>7.5</td>
<td>700</td>
<td>0.36</td>
<td>420</td>
<td>4.20</td>
<td>1.28</td>
</tr>
<tr>
<td>21.5</td>
<td>17</td>
<td>750</td>
<td>0.36</td>
<td>450</td>
<td>4.50</td>
<td>1.37</td>
</tr>
<tr>
<td>38.5</td>
<td>half-space</td>
<td>2200</td>
<td>0.36</td>
<td>1320</td>
<td>13.20</td>
<td>4.02</td>
</tr>
</tbody>
</table>

$^2$The value is obtained from the lab test data, which corresponds to 0.2% strain level.

The degradation of dynamic soil properties as observed in experiments has to be considered in finite element analysis in order to capture more accurate behaviors. As the shear wave velocity correlates with shear modulus by Equation 503.2,

$$v_{\text{shear}} = \sqrt{\frac{G}{\rho}}$$

we can readily obtain the dynamic degradation of wave velocities. This leads to smaller element size required for detailed simulation of wave propagation. The newly calculated element sizes are listed in Table 503.3. A three bent prototype finite element model has been developed with element size $\Delta h = 0.6m$ as shown in Figure 503.4.

503.2.1.3 Time Step Length Requirement

As stated in Preisig (2005), the time step $\Delta t$ used for numerically solving nonlinear vibration or wave propagation problems has to be limited for two reasons. The stability requirement depends on the numerical procedure in use and is usually formulated in the form $\Delta t = T_n < \text{value}$. $T_n$ denotes the smallest fundamental period of the system. Similar to the spatial discretization $T_n$ needs to be represented by about 10 time steps. While the accuracy requirement provides a measure on which higher modes of vibration are represented with sufficient accuracy, the stability criterion needs to be satisfied for all modes. If the stability criterion is not satisfied for all modes of vibration, then the solution may diverge. In many cases it is necessary to provide an upper bound to the frequencies that are present in a system by including frequency dependent damping to the model.

The second stability criterion results from the nature of the finite element method. As a wave front progresses in space it reaches one point after the other. If the time step in the finite element analysis is
too large the wave front can reach two consecutive elements at the same moment. This would violate
a fundamental property of wave propagation and can lead to instability. The time step therefore needs
to be limited to

$$\Delta t < \frac{\Delta h}{v}$$  \hspace{1cm} (503.3)

where $v$ is the highest wave velocity.

According to Table 503.3, the time step requirement can be obtained as

$$\Delta t < \frac{\Delta h}{v} = 0.00256$$  \hspace{1cm} (503.4)

in seconds.
503.2.1.4 Domain Reduction Method

Domain reduction method was originally proposed in Bielak et al. (2003b); Yoshimura et al. (2003b). The theory aims at reducing the size of simulation domain by means of variable interchange. This method features a two-stage strategy for complicated three-dimensional earthquake engineering simulations. The first is an auxiliary problem that simulates the earthquake source and propagation path effects with a model that encompasses the source and a background structure from which the localized feature has been removed. The second problem models local site effects. Its input is a set of equivalent localized forces derived from the first step. These forces act only within a single layer of elements adjacent to the interface between the exterior region and the geological feature of interest. The beauty of this theory comes from the fact that we can use established numerical and/or experimental approaches to solve the first-stage wave propagation problem. With the outcome of the first phase solution, we greatly reduce the size of the problem and then efforts can be focused on the second phase to deliver more accurate simulation on local responses. This approach can be successfully used in soil-foundation-structure-interaction finite element modeling without the need to incorporate unnecessary far-field motion simulations.

503.2.1.5 Structural Model

The nonlinear structure model developed in this work is a joint effort of UCB and UCD. Experimental data has been collected from UNR shaking table tests to calibrate the structural models. The effort in this work has been focused on how to integrate advanced structure model with geotechnical model to enable full-scale prototype simulations. The assumption that the plastic hinge forms either on the top of column or at the fixed bottom does not hold for SFSI problems. This restriction has been removed as the geotechnical and structural models are connected together.
Figure 503.5: Simplified Hinge Model Developed for SFSI Prototype Simulations (Dryden, 2005)
Figure 503.6: Moment-Rotation Relationship of Structural Hinge Model (Dryden, 2005)
Figure 503.7: Developed structural model (Dryden, 2005).
503.2.1.6 Simulation Scenarios

In order to fully investigate how the relative strength of soil to the structure can affect the system behaviors, a thorough parametric study has been performed. The prototype model used in this work is a 4-span bridge structure with 3 bents. The supporting soil foundation can be varied according to different site conditions. The mesh of the prototype finite element model is shown in the Figure 503.8. The underlying soil of the bridge can be soft bay mud or stiff sand. In order to fully investigate the SFSI response, various scenarios are simulated as shown in Table 503.4 and results are analyzed.
Figure 503.8: Finite Element Model for 3 Bent Prototype Bridge System
Table 503.4: Simulation Scenarios for Prototype SFSI Studies

<table>
<thead>
<tr>
<th>Simulation Cases</th>
<th>Soil Block 1</th>
<th>Soil Block 2</th>
<th>Soil Block 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
</tr>
<tr>
<td>Case 2</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
</tr>
<tr>
<td>Case 3</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
</tr>
<tr>
<td>Case 4</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
</tr>
<tr>
<td>Case 5</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
</tr>
<tr>
<td>Case 6</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
</tr>
<tr>
<td>Case 7</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
</tr>
<tr>
<td>Case 8</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
</tr>
</tbody>
</table>

Constitutive Modeling of Stiff Sand  For the stiff sand, the constitutive model developed in previous sections, as shown in Table 503.1 will be used.

Constitutive Modeling of Soft Clay  This soil model aims at simulating in-situ undrained behavior of soft bay mud. Undrained shear strength can be easily determined and a simple von Mises model is used in this research.

503.2.2 Earthquake Simulations - 1994 Northridge

Starting from this section, detailed numerical simulation results will be presented to show how the finite element simulation techniques can be used in prototype earthquake simulations. The results are presented here, and discussions will follow. Figure 503.9 shows the input motion recorded from 1994 Northridge which contains lots of high frequency contents. There are totally two motions are selected for this work, one with primary short period (high frequency) contents, and the other with primary long period (low frequency) contents. The purpose is to study every single component of the SFSI system trying to expose how each affects the SFSI system response.

503.2.2.1 Input Motion
Figure 503.9: Input Motion - Century City, Northridge Earthquake 1994
503.2.2 Displacement Response

This section presents the displacement results from numerical simulations.

There are a couple of interesting things that deserve attention. Firstly, in the 1994 Northridge earthquake which contains much high frequency content, the structural response from softer soil actually is smaller than those on top of stiff soil. This is interesting because it basically contradicts the common notion that stiffer the soil, stabler the structure. That is the case for static design. But for earthquake design, we are presenting different stories. Secondly, we see the soil displacement near structure is largely affected by the SFSI. So the question if it is valid to apply outcrop motion directly to fixity point to excite the structure, just as people commonly do, might need a revisit.

![Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 1)]
Figure 503.11: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 1)

Figure 503.12: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 2)
Figure 503.13: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 2)

Figure 503.14: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 3)
Figure 503.15: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 3)
503.2.2.3 Acceleration Response

The acceleration results are shown in this section, which also supports the observation that stiffer soil might not necessarily enhance the stability of the structure. Acceleration time series consistently show that the stiff soil will excite larger amplification for structures.

Figure 503.16: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 1)

Figure 503.17: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 1)
Figure 503.18: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 2)

Figure 503.19: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 2)
Figure 503.20: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 3)

Figure 503.21: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 3)
503.2.2.4 Displacement Response Spectra

It will be very interesting to look into the frequency domain what is going on. The structure is always stiffer than underlying soils. So if the underlying soil is stronger, it implies the natural frequency of the stiff soil will be closer to the structure on top of it.

If, the input motion contains much high frequency content, it will directly excite the stiffer soil so the structure on top will receive very large amplification. This conclusion is supported by following plots.

![Displacement Response Spectra](image)

Figure 503.22: Simulated Displacement Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 1)
Figure 503.23: Simulated Displacement Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 1)

Figure 503.24: Simulated Displacement Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 2)
Figure 503.25: Simulated Displacement Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 2)

Figure 503.26: Simulated Displacement Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 3)
Figure 503.27: Simulated Displacement Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 3)
503.2.2.5 Acceleration Response Spectra

The acceleration spectra have also been plotted to support the observation we made before. The consistent discovery is that stiffer soil will have amplification concentrated to the lower period side. If the input motion also have lower period contents, those will amplify the response the structure can see.

Figure 503.28: Simulated Acceleration Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 1)

Figure 503.29: Simulated Acceleration Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 1)
Figure 503.30: Simulated Acceleration Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 2)

Figure 503.31: Simulated Acceleration Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 2)
Figure 503.32: Simulated Acceleration Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases with Free Field Motions (Soil Block 3)

Figure 503.33: Simulated Acceleration Response Spectra, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 3)
503.2.6 Structural Response

Finally we come to the point that we can see exactly how the structure responds to excitation mechanically. The moment time series shown here tells some important stories during dynamic shaking. Firstly, the structure on top of stiffer soil will yield much faster than those in soft soils. This makes perfect sense after the observations we made in previous sections. The input motion contains very similar frequency content as the stiff soil so stiff soil and the structure on top of it are excited much more than the soft-soil-structure system. Secondly, The structure on top of soft soil will see larger residual response than the stiffer soil. This exactly tells the story that the soft soil will respond much largely to the long period content of input motions which is much closer to the natural period of the soft soil.

![Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 1 Pile 1)](image)

Figure 503.34: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 1 Pile 1)
Figure 503.35: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 1 Pile 2)

Figure 503.36: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 2 Pile 1)
Figure 503.37: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 2 Pile 2)

Figure 503.38: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 3 Pile 1)
Figure 503.39: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Eight Cases (Structure Bent 3 Pile 2)
503.2.3 Earthquake Simulations - 1999 Turkey Kocaeli

It has been well observed that the characteristics of input ground motion also affects the SFSI system response. From the perspective of soil dynamics, stiffer soil will have shorter natural period and thus higher natural frequency. One can argue that the conclusion that has been established in Section 503.2.4.1 might exactly reflect the case that for earthquake input motions containing much high frequency content, the stiff soil will always receive much stronger shaking. The 1994 Northridge earthquake has been known to contain very high frequency component as shown in Figure 503.40.

![Figure 503.40: Frequency Contents of Ground Motions - Acceleration Time Series](image)

So it will be also legitimate to question that if the ground motion has much long frequency (long period) content, the structure supported by soft soil might be the one to be exposed.

In this work, the question has been studied using the exact finite element models that we created in the previous sections.

The 1999 Turkey Kocaeli earthquake motion recorded at station Yarimca (YPT330) has been used as the target long period motion to study the SFSI behavior with different soil profiles.
Figure 503.41: Input Motion - Turkey Kocaeli Earthquake 1999
It has been shown in this work that long period (low frequency) motion will excites stronger response from softer clay soil layers which has a lower natural frequency as opposed to the conclusion we draw before in Section 503.2.4.1.

### 503.2.3.1 Displacement Response

This section is designed specifically to expose the effect of input motion on SFSI system responses. From the pictures shown below, you can see that now the structure on top of soft soil will show much larger response from the shaking. The story behind is obvious to explain. Now the input motion from Turkey Kocaeli earthquake contains primary long period content, which is similar to the natural frequency content of the soft soil. During shaking, this underlying resonance excites the response of the whole SFSI to a larger degree. While on the other hand, the stiff soil now is further away from the primary frequency of the input motion. The consequence is that now the stiff soil will not see much excitation, neither will the structure on top of it.

![Simulated Displacement Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 1)](image)

Figure 503.42: Simulated Displacement Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 1)
Figure 503.43: Simulated Displacement Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 1)

Figure 503.44: Simulated Displacement Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 2)
Figure 503.45: Simulated Displacement Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 2)

Figure 503.46: Simulated Displacement Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 3)
Figure 503.47: Simulated Displacement Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 3)
503.2.3.2 Acceleration Response

Acceleration plots might not show as clearly as those in displacement plots due to the reason that acceleration is derivative of displacement, so acceleration is more responsive to higher frequency contents than lower frequency contents. This effect actually can be observed from any recorded displacement and acceleration spectra records. The acceleration spectra will shift to the low period or high frequency side. This observation will be further explained in later sections when we discuss the moment time series.

Figure 503.48: Simulated Acceleration Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 1)
Figure 503.49: Simulated Acceleration Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 1)

Figure 503.50: Simulated Acceleration Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 2)
Figure 503.51: Simulated Acceleration Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 2)

Figure 503.52: Simulated Acceleration Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 3)
Figure 503.53: Simulated Acceleration Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 3)
503.2.3.3 Displacement Response Spectra

The response spectra are useful tools to look deep into the frequency domain. As we explained before, the Turkey Kocaeli shaking was picked for this study because it contains much longer period contents that were not present in the 1999 Northridge earthquake. In this section, we can clearly see how the soil foundation can affect the structures on top of it. Firstly, structures supported by stiff soil will only see amplification from those low period, high frequency components. Correspondingly, structures supported by soft soil tend to respond much more to long period, low frequency components of the shaking motion. Secondly, this plots exactly show how important the SFSI analysis is. It will not make much sense if one wants to analyze the structure without acknowledging the characteristics of the underlying soil foundation and the input motion. How the structure behaves is the combination of answers to many questions such as the stiffness of soil and the frequency contents of the motions.

Figure 503.54: Simulated Displacement Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 1)
Figure 503.55: Simulated Displacement Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 1)

Figure 503.56: Simulated Displacement Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 2)
Figure 503.57: Simulated Displacement Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 2)

Figure 503.58: Simulated Displacement Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 3)
Figure 503.59: Simulated Displacement Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 3)
503.2.3.4 Acceleration Response Spectra

As we discussed before, the acceleration response shows much focused content to the high frequency (low period) side as acceleration is the derivative of displacement. But it is still clearly shows the difference between the structures supported by stiff soil and those supported by soft soil. SFSI is crucial in the sense that the structure response must be determined by both short period and long period components. The overall response is determined by the primary periods of both underlying soil and the input motion.

Figure 503.60: Simulated Acceleration Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 1)
Figure 503.61: Simulated Acceleration Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 1)

Figure 503.62: Simulated Acceleration Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 2)
Figure 503.63: Simulated Acceleration Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 2)

Figure 503.64: Simulated Acceleration Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases with Free Field Motions (Soil Block 3)
Figure 503.65: Simulated Acceleration Response Spectra, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 3)
503.2.3.5 Structural Response

The structure response now directly shows how differently the structure can react to the same input motion, given different soil conditions. We can again see the structure on top of soft soil exhibits much response to long period content especially for the fact that the structure on top of the soft soil reaches maximum moment for about 3 seconds, which is not present in the stiff soil case. This is also consistent with the observation that the displacement of the structure supported by soft soil is much larger due to the much longer time for plastic slip when the plastic moment is reached.

Figure 503.66: Simulated Maximum Moment Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 1 Pile 1)
Figure 503.67: Simulated Maximum Moment Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 1 Pile 2)

Figure 503.68: Simulated Maximum Moment Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 2 Pile 1)
Figure 503.69: Simulated Maximum Moment Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 2 Pile 2)

Figure 503.70: Simulated Maximum Moment Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 3 Pile 1)
Figure 503.71: Simulated Maximum Moment Time Series, Turkey Kocaeli 1999, Yarimca, Comparison of Two Cases (Structure Bent 3 Pile 2)
Table 503.5: Simulation Scenarios for Prototype ESSI Studies

<table>
<thead>
<tr>
<th>Simulation Cases</th>
<th>Soil Block 1</th>
<th>Soil Block 2</th>
<th>Soil Block 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
</tr>
<tr>
<td>Case 8</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
</tr>
</tbody>
</table>

503.2.4 Earthquake Soil Structure Interaction Effects

In the following sections we analyze various Earthquake Soil Structure Interaction (ESSI) effects.

503.2.4.1 How Strength of Soil Foundations Affects ESSI

It has been well known that stiffer soil layer will provide higher bearing capacity of structures so site improvements will always be preferred in engineering practice when one talks about foundation design.

While for dynamic cases, this widely-held impression will not be valid anymore. In order to see how stiffness of soil can affect the response of the whole ESSI system during earthquake shaking, two distinct scenarios listed in Table 503.5, Case 1 with all stiff soil foundations, and Case 8 with all soft soil foundations have been extracted to show the dynamic system response.

Figure 503.72: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Structure Bent 1)

Case 1 and Case 8
Figure 503.73: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Structure Bent 2)

Figure 503.74: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Structure Bent 3)
Figure 503.75: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Structure Bent 1)

Figure 503.76: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Structure Bent 2)
Figure 503.77: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Structure Bent 3)
It has been shown in this research that if the soil layer is stiffer, structures will see much more amplification from earthquake shaking. Figures 503.72 to 503.74 show the acceleration response from ESSI systems with stiff sand and soft clay, which clearly shows that stiff sand delivers much stronger energy input to superstructures. The superstructures on top of stiffer soil also yield sooner than softer soil layers as shown in maximum moment time series Figures 503.75 to 503.77. This observation proves one interesting point that in order to improve structural stability, site improvement is not necessarily improving the dynamic resistance of the ESSI system. In later section of this work, further observation will be made to correlate this conclusion with the characteristics of the input motion.

As for the soil side, near-structure soil motion will be also affected by ESSI. So the traditional way of assuming that recorded ground motion can be used as input motion to ESSI analysis should be revisited. According to Figures 503.78 503.79 503.80, we can see that stiff soil also shows stronger surface motion records during earthquake shaking.

![Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Soil Block 1)](image)

Figure 503.78: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Soil Block 1)
Figure 503.79: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Soil Block 2)

Figure 503.80: Simulated Displacement Time Series, Northridge 1994, Century City, Comparison of Two Cases - First 25s (Soil Block 3)
Table 503.6: Simulation Scenarios for Prototype ESSI Studies

<table>
<thead>
<tr>
<th>Simulation Cases</th>
<th>Soil Block 1</th>
<th>Soil Block 2</th>
<th>Soil Block 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 3</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
</tr>
</tbody>
</table>

503.2.4.2 How Site Non-Uniformity Affects ESSI

Bridge is always constructed over gulfs or bays. It is a common case that for multi-span bridge, different bents will inevitably sit on soil foundations with totally different strength. This site nonuniformity complicates design because individual structure response might be largely varied.

In order to study the effects of site non-uniformity on the dynamic response of ESSI system, Case 3 listed in Table 503.6 has been selected as the test bed of our simulation. This scenario corresponds to the case that a 4-span bridge sits on solid abutments but with much softer bay-mud type foundation in the middle of a bay.

Figure 503.81: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

Case 3
Figure 503.82: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Figure 503.83: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
Figure 503.84: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

Figure 503.85: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)
Figure 503.86: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
Table 503.7: Simulation Scenarios for Prototype ESSI Studies

<table>
<thead>
<tr>
<th>Simulation Cases</th>
<th>Soil Block 1</th>
<th>Soil Block 2</th>
<th>Soil Block 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 2</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
</tr>
<tr>
<td>Case 5</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
<td>Stiff Sand</td>
</tr>
</tbody>
</table>

From acceleration Figures 503.81, 503.82, 503.83, and moment Figures 503.84, 503.85 and 503.86, we can clearly see that due to the presence of a soft soil block, all structures show smaller magnitude in response. This is especially true for the superstructure that directly sits on top of the softer soil foundation. For Case 3 listed in Table 503.6, the superstructure (bent 2) gets much smaller response because it is right on top of clay (bay mud) foundation. As a matter of fact, the middle bent (bent 2) does not yield at all.

We want to extend this observation to other cases as listed in Table 503.7. Case 2 shows the scenario that the soil foundation supporting bent 3 is soft bay mud, while Case 5 shows it for bent 1.

The same reasoning can be applied to these similar cases. Figures 503.87 to 503.92 show the results of Case 2 and Figures 503.93 to 503.98 for Case 5.

![Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)](image)

Figure 503.87: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

**Case 2**
Figure 503.88: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Figure 503.89: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
Figure 503.90: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

Figure 503.91: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)
Figure 503.92: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
Case 5
Figure 503.95: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)

Figure 503.96: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)
Figure 503.97: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Figure 503.98: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
Table 503.8: Simulation Scenarios for Prototype ESSI Studies

<table>
<thead>
<tr>
<th>Simulation Cases</th>
<th>Soil Block 1</th>
<th>Soil Block 2</th>
<th>Soil Block 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 4</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
</tr>
<tr>
<td>Case 6</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
<td>Soft Clay</td>
</tr>
<tr>
<td>Case 7</td>
<td>Soft Clay</td>
<td>Soft Clay</td>
<td>Stiff Sand</td>
</tr>
</tbody>
</table>

As a final conclusion, if there exists one substantially weaker soil layer for the ESSI system, the dynamic response for the whole ESSI system will be attenuated. The structure that directly sits on top of the soft soil block will receive significantly smaller excitation during earthquake shakings.

The other side of story can be also formulated using results obtained in this work. What if there is a substantially stronger soil foundation in the ESSI system? It would be worthwhile exploring the other side of reasoning. In this work, three other cases have been used to observe how the ESSI system will behave for the cases that one soil block is much stronger that the other blocks as listed in Table 503.8.

Figure 503.99: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

Case 4
Figure 503.100: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Figure 503.101: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
Figure 503.102: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

Figure 503.103: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)
Figure 503.104: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
It can easily be seen from Figures 503.99 to 503.104 that the structure on top of the strong soil block will exhibit much bigger response and while the other structures sitting on weaker soil blocks will not get much shaking at all.

Figure 503.105: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

Figure 503.106: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Case 6
Figure 503.107: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)

Figure 503.108: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)
Figure 503.109: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Figure 503.110: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
Figure 503.111: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)

Figure 503.112: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Case 7
Figure 503.113: Simulated Acceleration Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)

Figure 503.114: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 1)
Figure 503.115: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 2)

Figure 503.116: Simulated Maximum Moment Time Series, Northridge 1994, Century City, Comparison of Three Cases - First 25s (Structure Bent 3)
As a conclusion, dynamic behaviors of the ESSI system on non-uniform sites can be much more complicated than the uniform case. Generally speaking, the existence of soft soil foundation will attenuate the dynamic response of the whole ESSI system. The structure supported by soft soil block will see far less shakings than the ones supported by stiff soil. From the design perspective, the structure on top of strong soil foundations should be designed to higher safety with larger margin.

503.2.4.3 How Input Motion Affects ESSI

All the results have been shown in previous section, the purpose is to investigate how the whole ESSI system will respond to excitations with different predominant frequency.

From Figure 503.117 and 503.118, we can clearly see the difference in frequency contents of the two ground motions studied. The 1999 Turkey Kocaeli earthquake contains more long period components that are not present at the 1994 Northridge site.

![Figure 503.117: Frequency Contents of Ground Motions - Acceleration Time Series](image-url)
Figure 503.118: Frequency Contents of Ground Motions - Displacement Time Series
Depending on different site conditions, ESSI system will exhibit varied response to excitations with different predominant frequencies. As we can see from Figure 503.42 to Figure 503.47, the structure sitting on softer soil foundations now exhibits much larger structural response when it is subject to long period excitations. This conclusion has been supported by every single plot that records the response spectra of structures such as Figure 503.61, Figure 503.63 and Figure 503.65. For the long predominant periods present in the Kocaeli earthquake, the structures supported by soft clays exhibit much larger amplification than those supported by stiff soils.

It is also worthwhile to look deeper into the transfer function that tells some other aspects of the simulation scenarios. We can see from Figure 503.119 to Figure 503.130 that in terms of transfer function, which defines the amplification from soil surface to top of the structure, the structure is able to pick up much high frequency content that is not present within the soil layers. The visual representation of this is that the transfer function has much larger value at the high frequency end (low period end). This can be explained that typically people construct pile foundation inside the stiff soil (or bedrock). In our simulation, we also have a very stiff sustaining layer to provide pile tip resistance. The results shown in transfer functions make sense and also verifies our numerical model. Acceleration results are also shown as following.
Figure 503.119: Transfer Function of Simulated Displacement Time Series for Both Long Period and Short Period Motions (Structure Bent 1, from Soil Surface to Top of Bent)

Figure 503.120: Zoomed View: Transfer Function of Simulated Displacement Time Series for Both Long Period and Short Period Motions (Structure Bent 1, from Soil Surface to Top of Bent)
Figure 503.121: Transfer Function of Simulated Displacement Time Series for Both Long Period and Short Period Motions (Structure Bent 2, from Soil Surface to Top of Bent)

Figure 503.122: Zoomed View: Transfer Function of Simulated Displacement Time Series for Both Long Period and Short Period Motions (Structure Bent 2, from Soil Surface to Top of Bent)
Figure 503.123: Transfer Function of Simulated Displacement Time Series for Both Long Period and Short Period Motions (Structure Bent 3, from Soil Surface to Top of Bent)

Figure 503.124: Zoomed View: Transfer Function of Simulated Displacement Time Series for Both Long Period and Short Period Motions (Structure Bent 3, from Soil Surface to Top of Bent)
Figure 503.125: Transfer Function of Simulated Acceleration Time Series for Both Long Period and Short Period Motions (Structure Bent 1, from Soil Surface to Top of Bent)

Figure 503.126: Zoomed View: Transfer Function of Simulated Acceleration Time Series for Both Long Period and Short Period Motions (Structure Bent 1, from Soil Surface to Top of Bent)
Figure 503.127: Transfer Function of Simulated Acceleration Time Series for Both Long Period and Short Period Motions (Structure Bent 2, from Soil Surface to Top of Bent)

Figure 503.128: Zoomed View: Transfer Function of Simulated Acceleration Time Series for Both Long Period and Short Period Motions (Structure Bent 2, from Soil Surface to Top of Bent)
Figure 503.129: Transfer Function of Simulated Acceleration Time Series for Both Long Period and Short Period Motions (Structure Bent 3, from Soil Surface to Top of Bent)

Figure 503.130: Zoomed View: Transfer Function of Simulated Acceleration Time Series for Both Long Period and Short Period Motions (Structure Bent 3, from Soil Surface to Top of Bent)
Chapter 504

Earthquake-Soil-Structure Interaction, Nuclear Power Plants

(In collaboration with Dr. Nima Tafazzoli, Prof. José Abell, Dr. Yuan Feng, Mr. Sumeet Kumar Sinha, Dr. Han Yang and Dr. Hexiang Wang)
504.1 Stick/Solid Finite Element Model

The 3D finite element model is shown in Figure (504.1). The finite element model is created using both 8 node brick elements for modeling the soil part and 27 node brick elements for modeling the foundation part, and displacement beam element for structural components.

The size of the model is $140m$ along the X and Y directions, and $50m$ along the Z direction (the height of the soil layer). Shear wave velocity of the soil is considered to be $700m/s$ with density of $2200kg/m^3$. Foundation has the height of $5m$ embedded in the soil layer, with size of $90m$ in each horizontal direction. Concrete is chosen to have shear wave velocity of $2000m/s$ with density of $2400kg/m^3$. The size of the elements are $5m$ in each direction for both foundation and soil elements. Structural beam is composed of 12 displacement beam elements attached to each other with different stiffness and mass properties. Domain Reduction Method is used to apply the input effective forces. Both 1C and 3C wave propagation cases are using the same finite element model for second stage of DRM analysis.

Frictional contact/interface elements are placed at the interface of foundation and soil layer. The contact/interface element used here has the same normal and tangential stiffness with magnitude of $10^8 N/m$. Friction ratio and cohesion of the contact/interface element are $\mu = 0.4$ and $c = 0.0$ respectively. Contact elements are oriented along the Z axis of global model.

![Figure 504.1: 3D finite element SFSI model considering slipping behavior at the interface of foundation and soil layer](image)

In general it is accepted that in order to represent a traveling wave of a given frequency, the size of the finite elements have to have about 10 nodes per wavelength $\lambda$ (Hughes (1987); Argyris and Mlejnek...
(1991)). Using fewer than 10 nodes per wave length $\lambda$ specially for linear elements leads to numerical damping of higher frequencies as such element discretization misses certain frequencies of the wave. In order to satisfy this requirement, the size of the mesh should satisfy Equation (504.1) which is a function of the maximum frequency ($f_{\text{max}}$) of the input motion and also the shear wave velocity of the media $V_s = \sqrt{G/\rho}$ where $G$ is the shear modulus and $\rho$ is density of the soil.

$$\Delta h \leq \lambda/10 = V_s/(10 f_{\text{max}})$$  \hspace{1cm} (504.1)

In this model, the size of the elements are chosen to be $5m$ in each direction. Considering that the shear wave velocity of the soil is $700m/s$, the maximum frequency able to be propagated through this model is $14Hz$. 
504.1.1 Slipping behavior of SFSI models by considering 1C wave propagation

In this section, slipping behavior of SFSI systems is studied under 1C seismic wave propagation assumption to come up with the input motion to be used for the model prepared for second stage of DRM. Morgan Hill earthquake and Ricker wavelets are used as the source of motion.

Domain Reduction Method is used to apply the input effective forces. In order to obtain the displacement and acceleration time histories used for calculating the DRM layer effective forces, a finite element soil column with shear behavior is considered representing the 1C wave propagation. The motions are applied at the base of the 1C finite element model and propagated through the soil column using multiple support excitation pattern.
504.1.1.1 Morgan Hill earthquake

Figures (504.2) to (504.3) show the acceleration and displacement time histories and FFT of the Morgan Hill earthquake at the base of the 1D model. Figure (504.4) shows the acceleration time histories recorded at the bottom and top of the structure in X and Z directions.

As it is observed, the acceleration time history of slipping and no-slipping model has differences in amplitude and the phase in X direction of the model. However, the differences in phases of the recorded motions seems to be more along the Z direction. It shows that considering the slipping behavior has caused a lag in phase of the response. Comparison of displacement time histories of the same models is shown in Figure (504.5) along X and Z directions. As it is shown, displacement time histories along X direction are quite similar in this case while along Z direction they have different amplitudes due to initial settlement and different phases. FFT of the accelerations do not have much of a difference in terms of frequency content (shown in Figure (504.6) but they are different in amplitude which has been observed in acceleration time histories as well. In general it is observed that in this earthquake the the magnitude of accelerations and displacements are less at top of the structure comparing to the ones recorded at the bottom of the structure for both slipping and no-slipping behavior cases.

Figure (504.7) shows the distribution of sliding at the interface of foundation and the soil layer at 9 different time steps of analysis from 0.5 to 1.3 seconds. As it is observed, location of maximum sliding is changed along the foundation in time while magnitude and direction of the applied motion changes. Slipping happens at the specific steps and parts of the interface zone in which the applied force is more than the resistant one.

Figure 504.2: Acceleration and displacement time histories of Morgan Hill earthquake
Figure 504.3: FFT of Morgan Hill earthquake
Figure 504.4: Comparison of acceleration time histories of the structure between slipping and no-slipping models for Morgan Hill earthquake.
Figure 504.5: Comparison of displacement time histories of the structure between slipping and no-slipping models for Morgan Hill earthquake
Figure 504.6: Comparison of FFT of the acceleration of the structure between slipping and no-slipping models for Morgan Hill earthquake
Figure 504.7: Distribution of sliding along the contact/interface for Morgan Hill earthquake (gray scale given in meters)
504.1.1.2 Ricker wave

In order to investigate more, the Ricker wave with dominant frequency of $1/Hz$ is used for analysis of the same model. The maximum of this function happens at 1 second. Figures (504.8) and (504.9) show the acceleration and displacement time histories as well as the FFT of the Ricker wave respectively.

As shown in Figure (504.10), displacement time history of top of the structure along X direction is higher when slipping behavior is not considered, while along Z direction it is higher considering the slipping behavior. Since structural components have rotational degrees of freedom, the slipping behavior and gaps in the model will lead to have more rotational movement and rocking of the foundation. There is also a phase lag in response of the structure in slipping behavior model. The displacement time histories at the bottom of the structure are slightly different in X direction. It can also be observed that along Z direction, the response of the structure in slipping model is higher comparing to the no-slipping one.

In this analysis slipping behavior make the movement to be less along the X direction. This might be due to the fact that while the foundation and structure are moving, there are gaps created at some parts of the interface which will affect the rocking movement of the foundation.

Figure (504.11) shows the FFT of the acceleration at the bottom and top of the structure. As it is observed, there is a slight shift in predominant frequency of the response between the slipping and no-slipping behavior. By considering the slipping behavior, the dominant frequency of the motion is decreased along the X direction while is increased along the Z direction. This means that the system gets softer along X direction and stiffer along Z direction. This is due to the gap openings and slidings at the foundation and soil layer interface. Since sliding happens along the X direction, and also considering the gap openings occur along the Z direction which can lead to the rocking of the foundation, it makes the system softer along the X direction. In general this shows the fact that the natural frequency of the system can be changed due to the slipping behavior.

Figures (504.12) and (504.13) show the distribution of gap openings and slidings at the interface of foundation and the soil layer at 9 different time steps of analysis from 0.5 to 1.3 seconds. It can be observed from Figure (504.12) how the location of gap openings are changed on different parts of the contact/interface as the dynamic motion is applied. The location of gap openings are changed from one side to the other while the magnitude and direction of the applied motion changes. The maximum of the gap openings at these time steps happen at 1.3 second with maximum amount of $0.1m$ on the right side of the interface. Maximum sliding at the shown range of time steps happens at 1.0 second with maximum value of $0.07m$ on the left side of the interface.
Figure 504.8: Acceleration and displacement time histories of Ricker wave with dominant frequency of 1\(\text{Hz}\)

Figure 504.9: FFT of Ricker wave with dominant frequency of 1\(\text{Hz}\)
Figure 504.10: Comparison of displacement time histories of the structure between slipping and no-slippping models for Ricker wave
Figure 504.11: Comparison of FFT of the acceleration of the structure between slipping and no-slipping models for Ricker wave
Figure 504.12: Distribution of gap openings along the contact/interface for Ricker wave (gray scale given in meters)
Figure 504.13: Distribution of sliding along the contact/interface for Ricker wave (gray scale given in meters)
504.1.2 Slipping behavior of SFSI models by considering 3C wave propagation

In this section, the same finite element SFSI model as previous section is used with 3C wave propagated motions as input motions for simulations. The input motion used here is Ricker wave (as shown in Figure (504.8)).

In order to study the slipping behavior of SFSI considering 3C wave propagation for first stage of DRM simulation, a finite element model with dimensions of $10000m \times 50m \times 5000m$ is considered. Two cases are studied here with the source of motion (fault) to be located at $(x = 3000m, y = 0, z = 3000m)$ and $(x = 3000m, y = 0, z = 3000m)$. Figures (504.14) and (504.15) show these two cases respectively.

The size of the elements is chosen to be $50m$ in all directions for both cases in order to reduce the computational time. The soil parameters are: shear wave velocity of $700m/s$ and density of $2200kg/m^3$. Analyses for the fault slip model are done by applying the motion at the nodes of one element. This is done in order to represent the the wave propagation starting from the fault using multiple support excitation pattern. This is representing the first stage of analysis of DRM in which a big model including the fault is considered for free field case in order to obtain the required motions for DRM layer.

![Diagram](attachment://504.14.png)

Figure 504.14: Domain to be analyzed for the 1st stage of DRM with fault located at an angle of $45^\circ$ with respect to the top middle point of the model
Figure 504.15: Domain to be analyzed for the 1st stage of DRM with fault located at an angle of 34° with respect to the top middle point of the model
504.1.2.1 Ricker wave, with fault located at $45^\circ$ towards the top middle point of the model

For the first case to be studied here, Ricker wavelet is considered as an input motion with dominant frequency of $1Hz$. The fault is located with angle of $45^\circ$ toward the top middle point of the model (Figure (504.14)).

Displacement time histories of the structure along X and Z directions are shown in Figure (504.16). It is observed that along the X direction, the amplitude of displacement is little less at the beginning of the shaking of the slipping behavior model and a slight phase lag can be observed as well. However, when the actual pick of the motion is gone, it is shown that the motion for no-slipping model will damp out sooner. So the magnitude of displacements are higher at this time range for slipping model which could be because of the gap/sliding at the interface zone. The displacement time histories do not have a significant difference at the bottom of the structure. Displacement time histories observed along the Z direction shows the fact that magnitude of displacements for slipping model is higher and will damp out later comparing to the no-slipping model.

Acceleration time histories along X and Z directions at the bottom and top of the structure are shown in Figure (504.17). It is observed that along the X direction the amplitude of acceleration is less considering the slipping behavior and also there is a phase lag in the observed motion. However, the amplitude of the acceleration along Z direction is much higher in case of considering the slipping case. This is the same behavior observed in 1C wave propagation as well but with higher difference in time histories amplitudes.

As shown in Figure (504.18), the dominant frequency of the response in case of slipping is less than the one observed in no-slipping case along X direction while it is higher along Z direction.

Figures (504.19) and (504.20) show the distribution of sliding at the interface of foundation and the soil layer at 9 different time steps of analysis from 4.5 to 5.3 seconds. It can be observed from Figure (504.19) that maximum gap opening of $0.12m$ is occurred at 4.7 seconds while the location of the openings are changed during the analysis. In addition, maximum sliding at the interface zone in this case happens at 4.8 seconds with magnitude of $0.03m$. In both gap and slide distribution plots, it can be observed that the place of maximum is close to the middle of the foundation which is where the structure is located.

Distribution of cumulative dissipated energy due to sliding of the foundation and soil layer contact/interface zone is shown in Figure (504.21). By modeling the slipping behavior at the interface zone, part of the seismic energy is dissipated through the sliding and rocking of the foundation and therefore, less amount will be transferred to the structural components. Figure (504.22) shows how energy can be dissipated during the analysis for the point at the middle of the interface zone (location of the structure).
When sliding happens, some part of the energy is dissipated as shown while there will be no change in dissipated energy if the foundation and soil are sticking to each other.

Figure 504.16: Comparison of displacement time histories of the structure between slipping and no-slipping models for Ricker wave
Figure 504.17: Comparison of acceleration time histories of the structure between slipping and no-slipping models for Ricker wave
Figure 504.18: Comparison of FFT of the acceleration of the structure between slipping and no-slipping models for Ricker wave
Figure 504.19: Distribution of gap openings along the contact/interface interface for Ricker wave (gray scale given in meters).
Figure 504.20: Distribution of sliding along the contact/interface for Ricker wave (gray scale given in meters).
Figure 504.21: Distribution of cumulative dissipated energy due to sliding along the contact/interface for Ricker wave (gray scale given in kJ).

Figure 504.22: Cumulative dissipated energy time history due to sliding at the mid-center of the contact/interface for Ricker wave (gray scale given in kJ).
504.1.2.2 Ricker wave, with fault located at 34° towards the top middle point of the model

The last simulation to be studied here is the case which the location of the fault has an angle of 34° with respect to the top middle point of the model (Figure (504.15)). Same Ricker wave is used as an input motion to be propagated through the model built for first stage of DRM analysis. In this case, since the source of the motion is farther from the interested domain (comparing to previous case with the angle of 45°), motions will arrive later at the interface zone, and structure start shaking later.

Displacement time histories at the bottom and top of the structure are recorded and shown in Figure (504.23) along X and Z directions. Comparing to the previous case along X direction, at the beginning of the shaking the magnitude of the displacement for slipping behavior model is a bit smaller while it damps out sooner as well. Along the X direction, the trend of the displacement time histories are pretty much the same with a slight difference in magnitudes in time.

Acceleration time histories of the structure are shown in Figure (504.24) along X and Z directions. The same behavior is observed here as previous case such that the amplitude of the acceleration is significantly higher for case of no-slipping along the X direction while it is less along the Z direction. The seismic wave damps out sooner along X direction in case of considering the slipping behavior.

The frequency content change is shown in Figure (504.25) by comparing the FFT of the accelerations obtained from the model considering the slipping behavior and the one with no-slipping behavior. It can be observed that there is a slight change in frequency content and predominant frequencies along both X and Z directions.

Figure (504.26) shows the distribution of gap openings along the soil-foundation contact/interface zone for 9 time steps from 4.5 to 5.3 seconds. The maximum gap opening happens at 4.5 seconds with magnitude of 0.12m. On the other hand, the sliding at the contact/interface zone is shown in Figure (504.27) with maximum magnitude of 0.0035m happening at 4.5 seconds. In this case, the maximum sliding happens around the location of the structure.

Figures (504.28) and (504.29) show the distribution of the dissipated seismic energy at the interface zone and dissipated energy time history at the middle of the contact/interface zone respectively.
Figure 504.23: Comparison of acceleration time histories of the structure between slipping and no-slipping models for Ricker wave
Figure 504.24: Comparison of acceleration time histories of the structure between slipping and no-slipping models for Ricker wave
Figure 504.25: Comparison of FFT of the acceleration of the structure between slipping and no-slipping models for Ricker wave
Figure 504.26: Distribution of gap openings along the contact/interface for Ricker wave (gray scale given in meters)
Figure 504.27: Distribution of sliding along the contact/interface for Ricker wave (gray scale given in meters)
Figure 504.28: Distribution of cumulative dissipated energy due to sliding along the contact/interface for Ricker wave (gray scale given in kJ)

Figure 504.29: Cumulative dissipated energy time history due to sliding at the mid-center of the contact/interface for Ricker wave (gray scale given in kJ)
504.2 Three Dimensional (3C) Seismic Wave Fields and Behavior of Nuclear Power Plants (NPPs)

504.2.1 Development of Seismic Motions: Large Scale Free Field Model

- Large scale seismic free field
- Close up for large scale seismic free field

504.2.2 NPP Response, Model #01

- Free field at NPP location
- NPP response
- NPP response, cut-out of the model, inside response
504.3 3D Representative NPP Structure Model(s)
504.3.1 Model #01, Single NPP

Total number degrees of freedom (DOFs, unknowns) for Model 01 is 681,648. Table 504.3.1, gives other basic statistics for this model, while Figures 504.3.1 to 504.3.1 show disposition and views of the finite element mesh.

Table 504.1: Model 01 Statistics

<table>
<thead>
<tr>
<th>Components</th>
<th>Number of</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 DOF Nodes</td>
<td>205875</td>
</tr>
<tr>
<td>6 DOF Nodes</td>
<td>14413</td>
</tr>
<tr>
<td>27 node Bricks</td>
<td>23916</td>
</tr>
<tr>
<td>ANDES Shells</td>
<td>15627</td>
</tr>
<tr>
<td>Contact elements</td>
<td>2124</td>
</tr>
<tr>
<td>9 DOF Beams</td>
<td>583</td>
</tr>
</tbody>
</table>

Notes for Auxiliary Building:
- Interior walls except those highlighted above have a thickness = 2 ft.
- Exterior walls have a thickness of 3 ft.

Figure 504.30: Model01: Plan view (dimensions in feet).
Figure 504.31: Model01: Section view (dimensions in feet).

Figure 504.32: Model01: Soil layers.
Figure 504.33: Model01: Soil with the DRM (no slab).

Figure 504.34: Model01: Soil, with the slab and the DRM.
Figure 504.35: Model01: Slab only.

Figure 504.36: Model01: Auxiliary building
Figure 504.37: Model01: Containment building.

Figure 504.38: Model01: Full view.
504.3.2 Model #02, Single NPP

Simplified Representative 3D Model. Total number degrees of freedom (DOFs, unknowns) for Model 02 is 260,883. Table 504.2, gives other basic statistics for this model, while Figures 504.3.1 to 504.3.2 show disposition and views of the finite element mesh.

<table>
<thead>
<tr>
<th>Components</th>
<th>Number of</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 DOF Nodes</td>
<td>84820</td>
</tr>
<tr>
<td>6 DOF Nodes</td>
<td>3263</td>
</tr>
<tr>
<td>27 node Bricks</td>
<td>9576</td>
</tr>
<tr>
<td>ANDES Shells</td>
<td>8384</td>
</tr>
<tr>
<td>Contact elements</td>
<td>1249</td>
</tr>
<tr>
<td>9 Node Beams</td>
<td>490</td>
</tr>
</tbody>
</table>

Table 504.2: Model 02 Statistics

Figure 504.3.2 shows a general disposition of a representative 3D NPP model #02.

The model was developed as a cylindrical containment with dome top, while the auxiliary building surrounds the containment. There is no contact/interface between containment and the auxiliary building and current model has a gap of 0.2m while a new model (currently under development) will reduce this gap space, while still maintaining independence of two structural systems.
Finite element mesh for components and the complete system is shown in Figures 504.3.2 to 504.3.2.

Both structures (containment and auxiliary building) were placed on a slab foundation and then placed on a soil/rock base. Model is flexible enough that soil/rock properties and geology can be
varied. The sub-base model also includes a layer of elements for the DRM motion input, as well as two layers of elements outside the DRM layer for damping any outgoing waves. Models with a single NPP (Figure 504.3.2) is developed and used in analysis.
Figure 504.44: Model02: Containment building.

Figure 504.45: Model02: Auxillary and containment buildings.
Figure 504.46: Model02: Full view, Finite element model for a single NPP with the containment and auxiliary buildings on a common base mat, as well as the soil/rock sub-base, DRM layer for seismic motions input and the layers outside of DRM for damping out outgoing waves.
504.3.3 Model #03, Double NPP, Soil-Structure-Soil-Structure Interaction

Total number degrees of freedom (DOFs, unknowns) for Model 03 is 518,472. Table 504.3, gives other basic statistics for this model, while Figures 504.3.3 to 504.3.3 show finite element mesh.

<table>
<thead>
<tr>
<th>Components</th>
<th>Number of</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 DOF Nodes</td>
<td>168405</td>
</tr>
<tr>
<td>6 DOF Nodes</td>
<td>6526</td>
</tr>
<tr>
<td>27 node Bricks</td>
<td>19152</td>
</tr>
<tr>
<td>ANDES Shells</td>
<td>16768</td>
</tr>
<tr>
<td>Contact elements</td>
<td>2498</td>
</tr>
<tr>
<td>9 Node Beams</td>
<td>950</td>
</tr>
</tbody>
</table>

Figure 504.47: Model03: Soil, foundation slabs and the DRM layers.
Figure 504.48: Model03: Foundation slabs only.

Figure 504.49: Model03: Layered soil.
Figure 504.50: Model03: Auxillary buildings.

Figure 504.51: Model03: Containment buildings.
Figure 504.52: Model03: Auxiliary and containment buildings.

Figure 504.53: Model03: Full view.
Soil-Structure-Soil-Structure Interaction. Soil-Structure-Soil-Structure Interaction (SSSSI) need to be taken into account sometimes, as it might contribute to higher levels of seismic shaking for NPPs. It seems that in the case of making an assumption of elastic soil and rock beneath the NPP foundation, the SSSSI will have a larger effect most of the time, while with the inclusion of elastic-plastic models for soil and rock under the NPP foundation, those SSSSI influences will be reduced most of the time. This is illustrated in one of the models that was developed for analyzing SSSSI. While SSSSI for two NPPs (of similar stiffness and mass) conforms to the above observations, it is noted that SSSSI influences of smaller structures on larger (NPPs) can be mostly neglected, the SSSSI influence of larger structures (NPPs) on smaller structures probably cannot be neglected.

There are a number of ways to model SSSSI.

- Direct Models. The simplest and most accurate is to develop a direct model of both (two or more) structures on subsurface soil and rock, to develop input seismic motions and analyze results. While this approach is the most involved, it is also the most accurate, as it allows for proper modeling of all the structure, foundation and soil/rock geometries and material without making any unnecessary simplifying assumptions.

- Symmetry and Anti-Symmetry Models. These models are sometimes used in order to reduce complexity and sophistication of the direct model (see recent paper by Roy et al. (2013) for example). However, there are a number of concerns regarding simplifying assumptions that need to be made in order for these models to work. These models have to make an assumption of a vertically propagating shear waves and as such do not take into account surface waves (Rayleigh, Love, etc) that carry significant amount of seismic energy. These surface waves will additionally excite NPP for rocking and twisting motions, which will then be transferred to adjacent NPP by means of additional surface waves. If only vertically propagating waves are used for input (as is the case for symmetry and anti-symmetry models) energy of input surface waves is neglected. It is noted that depending on the surface wave length and the distance between adjacent structures, a simple analysis can be performed to determine if particular surface waves, emitted/radiated from one structure toward the other one (and in the opposite direction) can influence adjacent structures. It is noted that the wave length can be determined using a classical equation $\lambda = v/f$ where $\lambda$ is the length of the (surface) wave, $v$ is the wave speed\(^1\) and $f$ is the wave frequency of interest. Table 504.4 below gives Rayleigh wave lengths for four different wave frequencies (1, 5, 10, 20 Hz) and for three different Rayleigh (very close to shear) wave velocities (300, 1000, 2500 m/s):

\(^1\)For Rayleigh surface waves, their speed is just slightly below the shear wave speed (within 10%, depending on elastic properties of material), so a shear wave speed can be used for making these Rayleigh wave length estimates.
Table 504.4: Rayleigh wave length as a function of wave speed [m/s] and wave frequency [Hz].

<table>
<thead>
<tr>
<th>Wave Speed [m/s]</th>
<th>1.0Hz</th>
<th>5.0Hz</th>
<th>10.0Hz</th>
<th>20Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>300m/s</td>
<td>300m</td>
<td>60m</td>
<td>30m</td>
<td>15m</td>
</tr>
<tr>
<td>1000m/s</td>
<td>1000m</td>
<td>200m</td>
<td>100m</td>
<td>50m</td>
</tr>
<tr>
<td>2500m/s</td>
<td>2500m</td>
<td>500m</td>
<td>250m</td>
<td>125m</td>
</tr>
</tbody>
</table>

It is apparent that for given separation between NPP buildings, different surface wave (frequencies) will be differently transmitted with different effects. For example, for an NPP building that has a basic linear dimension (length along the main rocking direction) of 100m, the surface wave the low frequency waves (1Hz) in soft soil ($v_s \approx 300$ m/s) will be able to encompass a complete building within a single wave length, while for the same soil stiffness, the high frequency (20Hz) will produce waves that are too short to efficiently propagate through such NPP structure. On the other hand, for higher rock stiffness ($v_s \approx 2500$ m/s), waves with frequencies all the way up to approximately 5Hz and maybe even 10Hz, have an extent that can easily be affecting a building with a 100m dimension.

Further comments on symmetric and antisymmetric models:

- Symmetry: motions of two NPPs are out phase and this represents an unrealistic case, unless the wave length of surface wave created by one NPP (toward the other NPP) is so large that half wave length will encompass both NPPs. This type of motions (symmetry) is illustrated in figure 504.3.3 below

![Symmetric mode](image)

Figure 504.54: Symmetric mode of deformation for two NPPs near each other.

- Antisymmetry: motions of two NPPs are in phase and while that is more realistic than the symmetry case, still requires perfect matching of 1C input motions and the soil/rock conditions beneath, and as such is not realistic. This type of SSSSI is illustrated in figure 504.3.3 below
Figure 504.55: Anti-symmetric mode of deformation for two NPPs near each other.
504.3.4 Model #04, Small Modular Reactor (SMR)

Figure 504.56: Small Modular Reactor model, top (surface) view.

Figure 504.57: Small Modular Reactor model, cross section (half model) view.
504.4 3C (6C) vs 1C Seismic Motions

Realistic seismic wave fields are fully three dimensional, consisting of a body and surface waves, and featuring translational and rotational components of motions at and near the surface. However, current focus of various seismic analysis standards is on 1C motions. These 1C motions are representing one component of a full 3C wave field. More recently, use of $3 \times 1$C wave fields have been advocated, where 1C wave motions are replicated in other horizontal and a vertical direction with certain scaling factors. Use of $3 \times 1$C motions makes an implicit assumption that vertical motions are resulting from 1C compressional waves. This assumption is usually not true, as most of the vertical motions are a result of surface wave motions.

504.4.1 Appropriate Use of 3C and $3 \times 1$C and 1C Seismic Motions

We start by pointing out one of the biggest simplifying assumptions made, is that of a presence and use of 1C seismic waves. As pointed out in section 4.2.1 above, worldwide records do not show evidence of 1C seismic waves. It must be noted, that an assumption of neglecting full 3C seismic wave field and replacing it with a 1C wave field can sometimes be appropriate. However, such assumption should be carefully made, taking into account possible intended and unintended consequences.

A brief discussion on 1C, $3 \times 1$C and 3C seismic wave modelling and effects on SSI is provided below:

- 1C modelling of seismic waves is possible if material modelling for soil is linear or equivalent linear elastic. In this case, 1C motions from different directions (horizontal) can be combined, as superposition principle applies for linear elastic systems (soil in this case). Modeling of vertical motions using 1C approach is abit different as an analysis needs to be performed to decide if the vertical wave is a compressional wave (primary, P wave) or if vertical motions are a consequence of vertical components of surface waves. More on those options is provided below in $3 \times 1$C modelling option.

- $3 \times 1$C modeling of seismic waves is possible, similarly to the above case, if soil material is linear or equivalent linear elastic. SAs noted above, superposition principle can be applied and motions from each direction can be superimposed to obtained 3C motions at the surface. Since most of the time vertical motions are a results (consequence) of Rayleigh surface waves, it is important to analyze vertical motions and decide if modelling motions as 1C is appropriate. To this end, a wave length of surface wave plays an important role. If the Rayleigh surface wave length (which features both horizontal and vertical components) is longer than 12 times the dimension of the object (NPP), than object rotations, due to differential vertical displacements at object ends, are
indeed fairly small and object does move up and down as if excited with a vertical wave. This is shown in Figure 504.58 as the upper case. On the other hand, if the wave is long less than 12 object dimensions, then vertical motions are gradually replaced by object rotations, while vertical motions are reduced. Case in the lower left corner of Figure 504.58 shows a limiting case where seismic wave is 4 times longer than object dimension, which results in minimal vertical motions of the object, and maximum rotations, due to differential motions of object ends. For shorter surface waves, as shown in Figure 504.58, lower right case, waves might not even be exciting any significant dynamic behavior of the object (except local deformation) as their wave lengths are shorter than twice object length.

- 3C modelling, when done properly will capture all the body and surface wave effects for SSI analysis of NPPs.

![Figure 504.58: 1C vs 3×1C vs 3C. Three different cases of surface wave wave length. Upper case is where the surface wave length is 12 or more times longer than the object (NPP) dimension). Lower left case is where the surface wave length is only four times longer than the wave length, and lower right case is where the surface wave length is only two times longer than the object length.](image)

### 504.4.2 Illustration of Use of 3C and 1C Seismic Motions

A simple example can be used to illustrate differences in 1C vs 3C seismic motions. Assume that a full 3C (6C, 3 translational components and 3 rotational components) motions at the surface are only recorded in one horizontal direction. From a 1C recorded component one can develop a vertically propagating shear wave in 1C, that exactly models 1C recorded motion. This is usually done using de-convolution Kramer (1996a). Figure 504.59 illustrates the idea of using a full 3C seismic wave field to develop a reduced, 1C wave field.
Two seismic wave fields, the original wave field and a subset 1C wave field now exist. The original wave field includes body and surface waves, and features translational and rotational motions. On the other hand, subset 1C wave field only has one component of motions, usually an SV component (vertically polarized component of S (Secondary) body waves).

Figure 504.61 shows a snapshot of a full 3C wave field, resulting from a large scale regional simulation, from a point source (simplified), propagating P and S waves through layers.

Figure 504.62 shows a snapshot of the same wave field as in Figure 504.61, now focused on an immediate vicinity of location of interest, where blue stick is positioned.

It should be noted that regional simulation model shown in Figures 504.62 and 504.61 is rather simple, consisting of a point source at shallow depth in a 3 layer elastic media. Waves propagate, refract at layer boundaries (turn more "vertical") and, upon hitting the surface, create surface waves (in this case, Rayleigh waves). In our case (as shown), out of plane translations and out of plane rotations are not developed, however this simplification will not affect conclusions that will be drawn. A seismic wave field with full 3 translations and 3 rotations (6C) will only emphasize differences that will be shown later.

Figures 504.63 and 504.64 show local free field model with 3C and 1C wave fields respectively.

Please note that seismic motions are input in an exact way, using the Domain Reduction Method Bielak et al. (2003a); Yoshimura et al. (2003a) and how there are no waves leaving the model out of DRM element layer (4th layer from side and lower boundaries). It is also important to note that horizontal motions in one direction at the location of interest (in the middle of the model) are exactly the same for both 3C motions case and for a 1C motions case.

Figure 504.65 shows a snapshot of an animation (available through a link within a figure) of difference
in response of an NPP excited with full 3C (6C) seismic wave field, and a response of the same NPP to 1C seismic wave field.

Figures 504.66 and 504.67 show displacement and acceleration response on top of containment building for both 3C and 1C seismic wave fields.

A number of remarks can be made:

- Accelerations and displacements (motions, NPP response) of 6C and 1C cases are quite different. In some cases 1C case gives bigger influences, while in other, 6C case gives bigger influences.

- Differences are particularly obvious in vertical direction, which are much bigger in 6C case.

- Some accelerations of 6C case are larger that those of a 1C case. On the other hand, some displacements of 1C case are larger than those of a 6C case. This just happens to be the case for given source motions (a Ricker wavelet), for given geologic layering and for a given wave speed (and length). There might (will) be cases (combinations of model parameters) where 1C motions model will produce larger influences than 6C motions model, however motions will certainly again
Figure 504.62: Snapshot of a full 3C wave field, with body and surface waves, resulting from a point source at 45° at depth, down-left. This is a large regional scale model of a (simplified) fault with soil layers. Figure is a link to an animation of a full wave propagation.

be quite different. There will also be cases where 6C motions will produce larger influences than 1C motions. These differences will have to be analyzed on a case by case basis.

In conclusion, response of an NPP will be quite different when realistic 3C (6C) seismic motions are used, as opposed to a case when 1C, simplified seismic motions are used.
Figure 504.63: Snapshot of a full 3C wave field at the location of interest (where an NPP will be founded), featuring body and surface waves.

Figure 504.64: Snapshot of a reduced 1C wave field at the location of interest (where an NPP will be founded), featuring just SV body waves.
Figure 504.65: Snapshot of a 3C (6C) vs 1C response of an NPP, upper left side is the response of the NPP to full 3C wave field, lower right side is a response of an NPP to 1C wave field.

Figure 504.66: Displacements response on top of a containment building for 3C and 1C seismic input.
Figure 504.67: Acceleration response on top of a containment building for 3C and 1C seismic input.

Figure 504.68: Displacements response at the top corner of auxiliary for 3C and 1C seismic input.
Figure 504.69: Acceleration response at the top corner of auxiliary for 3C and 1C seismic input.
504.5 3C (6C) vs 3 × 1C vs 1C Seismic Motions

This section is from our paper (Abell et al., 2018).
504.6 3D Nonlinear Modeling for Nuclear Power Plants

This section is based on Sinha et al. (2017).

It noted that input files for these models are available at this LINK, and can be directly simulated using Real-ESSI Simulator, http://real-essi.us/, that is available on Amazon Web Services, https://aws.amazon.com/.

504.6.1 Introduction

Seismic simulations to structures are often done by 1-D input excitations defined from a family of damped response spectra. These input motions are applied uniformly to the entire base of the structure regardless of its dimension and dynamic characteristics of the soil, foundation and motion itself. This not only ignores the foundation and its contact/interface with soil, soil-structure interaction (SSI) but also the 3C nature and variability of seismic waves.

Interest to study SSI effects has grown significantly in recent years. However Tyapin (2007) and Lou et al. (2011) note that even after four decades of extensive SSI research, there still exists a large gap. Lou et al. (2011) notes that spatial analysis of full model in 3D is hardly done. To reduce the amount of calculations, many existing publications simplify extremely the super-structure to spring mass damper model or consider only limited interaction. Elgamal et al Elgamal et al. (2008) performed a 3D analysis of a full soilbridge system, focusing on interaction of liquefied soil in foundation and bridge structure. Jeremić et al. (2009) showed a full 3D soil-structure interaction of a prototype bridge, devised as a part of grand challenge, pre-NEESR project.

Investigations of SSI have shown that the dynamic response of a structure supported on elastic-plastic soil may differ significantly from the response of the same structure when supported on a rigid base Chopra and Gutierrez (1974); Bielak (1978). The difference comes because of the dissipation of part of the vibrational energy (seismic energy) by hysteresis action of the soil or structure itself. This results in damping of high frequency components, which could potentially prove quite useful for equipment that are prone to damage from high frequencies. On the other hand Jeremić et al. (2004) found that SSI can have detrimental effects on structural behavior as well and is dependent on the dynamic characteristics of the earthquake motion, the foundation soil and the structure.

Dissipation of energy during seismic events is another important factor to consider in design for its safety and economy. Dissipating energy in structure can lead to material degradation and damage. It is desired to dissipate most of the energy in soil with acceptable level of deformations in structure. A common neglect of plastic free energy has been observed in many publications, which results in clear violation of the second law of thermodynamics. A thermomechanical framework that can correctly
evaluate energy transformation and dissipation in dynamic SSI simulation was presented by Yang et al. (2018, 2019) based on works of Rosakis et al. (2000); Dafalias et al. (2002). This framework is applied to the prototype NPP model that is being analyzed in this paper. Locations with high possibility of damage are identified and insights on design improvement are discussed.

Only a few full 3D SSI interactions have been studied that too mainly focusing on bridges or small soil-foundation system. However, as per authors knowledge a full 3D non-linear analysis for a structure with soil-foundation-structure and contact/interface effects have not been investigated. Purpose of this paper here is to present a methodology for high fidelity modeling of seismic soil-foundation-structure (SFSI) interaction for a prototype of Nuclear Power Plant (NPP) with surface (shallow) foundation. Presented methodology employs the currently best available models and simulation procedures. In addition to presenting such state-of-the-art modeling, simulation results are used to illustrate non-linear-effects on seismic response of a prototype NPP model.

### 504.6.2 Model Development and Simulation Details

The Nuclear Power Plant (NPP) modeled here is a symmetric structure with shallow foundation of thickness 3.5m and size 100m. Figure 504.70 shows a slice view of the model in normal y direction (perpendicular to plane of the paper). Solid brick elements were used to model soil and foundation. The NPP structure was modelled by elastic shell elements. This section describes the material and modeling parameters summarized in Table 504.5, foundation, structure, and contact. Given also is a brief description of staged loading and seismic force application using domain reduction method (DRM).

![Diagram of Nuclear power plant model with shallow foundation.](image-url)
504.6.2.1 Structure Model

The NPP structure consists of auxiliary building, containment building and shallow foundation as shown in Figure ???. The auxiliary building consists of 4 floors of 0.6m thickness, ceiling floor of 1m thickness, exterior wall of 1.6m thickness and interior walls of 0.4m thickness. The exterior and interior walls are embedded down to the depth of the foundation. The containment building is a cylinder of diameter 20m and height 40m with wall thickness of 1.6m. There is a gap of 0.2m between the containment and auxiliary building. Top of the containment building is covered by semi-spherical dome of radius 20m. The foundation is square shallow footing of size 100m and thickness 3.5m. The containment building and the auxiliary building were modelled as shell elements and foundation as linear brick elements, both having the properties of concrete of elastic Youngs modulus 20GPa, poisons ratio 0.21 and density 2400kg/m³. The containment building which is more flexible than the auxiliary building had its first mode as bending with fundamental frequency at 4Hz.

504.6.2.2 Soil Model

The depth of the soil modelled below the foundation was 120 m, which is also the depth of DRM layer Sec 504.6.2.5. It is assumed that within this range the soil will plastify because of its self-weight, structure and seismic motions. The soil is assumed to be a stiff saturated-clay with undrained behavior having shear velocity of 500 m/s, unit weight of 21.4 kPa and Poisson’s ratio of 0.25. To represent the travelling wave accurately for a given frequency, about 10 nodes per wavelength i.e. about 10 linear or 3 quadratic brick elements are required. Here, the seismic waves are analyzed up to $f_{max} = 10Hz$. The smallest wavelength $\lambda_{min}$ to be captured thus, can be estimated as

$$\lambda_{min} = \frac{v}{f_{max}}$$

(504.2)
where, $v$ is the smallest shear wave velocity of interest. For $v = 500\, m/s$ and $f_{\text{max}} = 10\, Hz$ the minimum wavelength $\lambda_{\text{min}}$ would be $(500\, m/s)(10/\, s) = 50\, m$. Choosing 10 nodes/elements per wavelength the element size would be 5m. Jeremić et al. (2009); Watanabe et al. (2017) state that even by choosing mesh size $\Delta h = \lambda_{\text{min}}/10$, smallest wavelength that can be captured with confidence is $\lambda = 2\Delta h$ i.e. a frequency corresponding to $5f_{\text{max}}$. Based on the above analysis, soil was modeled as linear 8-node brick elements with grid spacing of $\Delta h = 5\, m$.

Because of the complex plastic-behavior of the soil many sophisticated models Yang et al. (2003); Dafalias and Manzari (2004b) have been developed to capture the non-linear response of soil. Wair et al. (2012) provides an empirical correlation to predict the shear strength of soil for given shear velocity $V_s$. Dickenson (1994) proposed the following relationship Eq 504.3 between $V_s$ and undrained strength $S_u$ for cohesive soils in San Francisco Bay Area.

$$V_s[m/s] = 23(S_u[kPa])^{0.475}$$  \hspace{1cm} (504.3)

Thus, for $V_s = 500\, m/s$, the undrained strength $S_u$ would be $650\, kPa$. Here, two scenarios of soil properties were considered in analysis. One linear elastic and the other as von-Mises with non-linear kinematic hardening of Armstrong Frederick type. For $S_u$ of $650\, kPa$ and $E = 1.3\, Gpa$, the non-linear inelastic model was calibrated for yield strength achieved at $0.01\%$ shear strain with linear kinematic hardening rate ($h_a$) as $30\, MPa$ and non-linear hardening rate ($c_r$) as 25. The soil properties is summarized in Table 504.5. The stress-strain response for the non-linear material model is shown in Figure 504.72.

![Stress-strain response](image)

Figure 504.72: Soil and contact/interface modeling.

### 504.6.2.3 Interface/Contact Modeling

Node-to-node penalty based soft contact/interface element Sinha and Jeremić (2017) was used to model the interaction between foundation and soil as they are not one continuum material. In soft contact, normal contact/interface force $F_n$ from soft-soil is assumed to increases exponential with penetration $\delta_n$.
as shown in Eq 504.4. The normal force $F_n$ and stiffness $K_n$ in defined as

$$F_n = k_{n,\text{init}} \times \exp(S_r \times \delta_n) \times \delta_n$$

$$K_n = \max(k_{n,\text{init}} \times \exp(S_r \times \delta_n) \times (1 + k_{n,\text{init}} \times \delta_n), k_{n,\text{max}})$$

(504.4)

where $\delta_n$ refers to the relative displacement between contact/interface node pairs in normal contact direction, $k_{n,\text{init}}$ refers to the normal stiffness in normal contact/interface direction, $S_r$ refers to the stiffening rate in normal contact direction. $k_{n,\text{max}}$ refers to maximum normal stiffness and provides a cap on exponentially increasing stiffness to make the solution numerically stable. The soft contact/interface was implemented to capture the phenomenon of increasing stiffness of soil with increasing penetration. Figure ?? shows the stiffness curve with penetration for the chosen contact/interface parameters also shown in Table 504.5.

Contact elements were applied all around the foundation connecting to the soil as shown in Figure ?? in red color zone. To ensure the stability of the numerical solution, the penalty stiffness in normal direction was chosen $2-3$ order magnitude greater than the stiffness of the soil. The Coulomb's friction coefficient $\mu$ between the soil and the foundation was chosen as 0.25. Viscous damping of 100Ns/m in normal and tangential damping was provided to model viscous damping arising from water.

### 504.6.2.4 Seismic Motions

3C seismic motions were developed by Rodgers (2017) using SW4 (Serpentine Wave Propagation of 4th order) Petersson and Sjögreen (2018) for an earthquake of magnitude ($M_w$) of 5.5 modelled with a point source on a fault of dimension $5.5 \times 5.6km$ with up-dip rupture slip model. The ESSI (Earthquake Soil Structure Interaction) box to capture the free-field motion was located on the foot-wall of the reverse thrust fault. The generated motion had a directivity effect as the fault slips and propagates in $x$-direction. Also, since the ESSI box was located perpendicular to the fault, strong motions in $y$-direction was expected.

Acceleration and displacement time-series of the motion at the center of ESSI box is shown in Figure 504.86. The peak ground acceleration (PGA) in $x$ and $y$ direction is about $0.5g$. Significant amount of vertical motions PGA of $0.2g$ can be observed which is neglected in many conventional seismic simulations. Since, the fault is located at foot-wall side of reverse thrust fault, there is permanent subsidence of about $50mm$ in $z$-direction at the end of shaking event. Fourier transform and response spectrum of the motions are shown in Figure 504.87. The frequency range of the motion is within $20Hz$. Response spectrum plot shows amplification for natural frequency greater than $2Hz$. Since, many equipments in nuclear industry operate at high frequencies, determination of high frequency excitation of NPP building is critical for design.
504.6.2.5 Domain Reduction Method

Domain Reduction Method (DRM) Bielak et al. (2003a) was used to apply 3C seismic motions generated from SW4 all around the model as shown in Figure ???. DRM is one of the best methods that can apply free field 3C ground motions to a finite-element model. It features a two-stage strategy for a complex, realistic 3D earthquake engineering simulation. First, is the generation of free field model with correct geology and second is the application of the generated free-field to the structure of interest. The DRM layer here is modeled as a single layer of elastic soil. Three damping (absorbing) layers adjacent to DRM layer were modeled to prevent incoming of reflected waves. For this analysis, 60\% Rayleigh damping was applied in each of the damping and DRM layers. The Rayleigh damping was applied in the frequency range of 1-5Hz.
504.6.2.6 Staged Simulation

The whole analysis was simulated with two loading stages. First stage was static self-weight to get the initial stress state of the soil and contact/interface elements. In second stage, seismic motion was applied using DRM method. For each stage, equilibrium was achieved using full Newton-Raphson method with a small tolerance of $1e^{-4}N$ on second-norm of unbalanced force. For dynamic analysis, Newmark integration method with numerical damping $\gamma = 0.7$ was used. Rayleigh damping of 2% in structure and 30% in soil was applied. The time step considered here was 0.02 seconds with simulation running in total for 40 seconds.

The analysis was run in parallel in Real-ESSI Simulator Jeremić et al. (1988-2021), http://real-essi.info. on eight CPUs. The model consisted of about $300k$ degrees of freedoms (dofs). Four scenarios (a) elastic no contact/interface (b) elastic with contact (c) elastic-plastic no contact and (d) elastic-plastic with contact/interface were performed. In this paper, unless specified elastic means elastic without contact/interface and inelastic means elastic-plastic with contact.

504.6.3 Simulation Results

Due to the space restriction, only few locations are selected to study the non-linear effects on NPP structure. The selected locations are shown in Figure 504.75. Since the containment building is more flexible than auxiliary building, location (D) in Figure 504.75 located on the top of the containment building is naturally the point of interest as it describes maximum drift during shaking. Three locations (A), (B) and (C) located at center of foundation is also selected to study the slip at interface during shaking.

![Figure 504.75: Locations selected to study non-linear effects and plot of total displacement at center of model Elastic (elastic with contact) and Inelastic (Elastic-Plastic with contact).](image)

Since the site is located on the foot wall, during seismic shaking the whole structure along with soil subsides down by about 50mm in elastic and 100mm in inelastic case. Overall, if self-weight stage is
also included, the soil settles by $150 \text{mm}$ in elastic and $350 \text{mm}$ in inelastic case as shown in Figure ??.

It is important to predict the development of high frequency excitation during shaking because it can prove to be alarming (when close to fundamental frequency) for nuclear-equipment. These high frequencies are thus, important to be monitored, predicted during earthquakes, for design of nuclear building to ensure the safety of equipment. Figure 504.76 plots the acceleration and its Fourier amplitude for the location (D). It is interesting to observe, the elastic-plastic analysis kills high frequency excitations in the structure which are persistent in elastic analysis. Elastic-plastic soil shows natural damping to some high frequencies because of dissipation of energy in form of heat by hysteresis loop. This can prove to be bigly useful for safe operation of nuclear equipments even at strong seismic events. The effects of contacts coupled with elastic-plastic material leads to huge dissipation of energy reducing the high frequency modes. In Z-direction, very little significant excitation was observed.

![Graphs showing accelerations and Fourier amplitudes for elastic and inelastic cases.](image)

Figure 504.76: Seismic response at top of containment building Elastic (elastic without contact) and Inelastic (elastic-plastic with contact).

The introduction of contact/interface can result in opening and closing of gaps at the soil-foundation interface for stronger earthquakes. However, here for the considered seismic motion for both elastic and inelastic case with contact, no uplift was observed. Figure 504.77 shows the relative displacement of NPP structure for elastic and inelastic analysis at 11 seconds. In elastic case, the structure drifts a lot while the deformation in soil remains small. Whereas, in the inelastic case, the soil deforms and plastify in z-direction keeping the structure deformation small. Thus the elasto-plastic soil acts as a natural base isolators material. This demonstrates that for the considered earthquake motions, the elastic-plastic soil can prove to quite beneficial because of small deformation and excitation in structure.
Figure 504.77:  Deformation of the NPP structure at 11 seconds (scaled 100 times).

Figure 504.78 plots the inter-slip of foundation with respect to soil at location (A), (B) and (C) for elastic and elastic-plastic case with contact. It can be observed that point (A) and (B) slips both relatively towards each other describing the presence of surface waves. The center of the foundation (B) bends and slides comparatively less than the exterior ends. This also strongly shows the directivity effects of the motion coming from the $-x$ to $+x$ direction. The directivity effect is more pronounced in inelastic analysis resulting in comparatively more slip and permanent deformation. Careful observation of sliding in $x$-direction, shows a permanent slip of $18\,\text{mm}$ for elastic-plastic soil. Although not shown in figure, the whole NPP structure show tendency of rotation about its center of mass during the DRM stage.

Figure 504.78:  Slip of foundation with respect to soil beneath it in $x$-$y$ slice plane for Elastic (elastic with contact) and Inelastic (Elastic-Plastic with contact).
Figure 504.79: Animation of a linear elastic vs inelastic response of an NPP.

Figure 504.80: Free Field.
504.6.4 Energy Dissipation

The energy dissipation in decoupled elastic-plastic material under isothermal condition is given by Yang et al. (2019):

\[ \Phi = \sigma_{ij} \dot{\varepsilon}_{ij} - \sigma_{ij} \dot{\varepsilon}_{ij}^{el} - \rho \dot{\psi}_{pl} \geq 0 \]  

(504.5)

where \( \Phi \) is the rate of change of energy dissipation per unit volume (or dissipation density), \( \sigma_{ij} \) and \( \varepsilon_{ij} \) are the stress and strain tensors respectively, \( \varepsilon_{ij}^{el} \) is the elastic part of the strain tensor, \( \rho \) is the mass density of the material, and \( \psi_{pl} \) is the plastic free energy per unit volume (or plastic free energy density). Note that Equation 504.8 is derived from the first and second laws of thermodynamics, which indicate the conditions of energy balance and nonnegative rate of energy dissipation, respectively.

Considering all possible forms of energy inside SSI system, the energy balance between input mechanical work \( W_{\text{Input}} \) and the combination of internal energy storage \( E_{\text{Stored}} \) and energy dissipation \( E_{\text{Dissipated}} \) can be expressed as:

\[ W_{\text{Input}} = E_{\text{Stored}} + E_{\text{Dissipated}} = KE + SE + PF + PD \]  

(504.6)

where \( KE \) is the kinetic energy, \( SE \) is the elastic strain energy, \( PF \) is the plastic free energy, and \( PD \) is the energy dissipation due to material plasticity. Equations for each energy component can be found in Yang et al. (2018). Note that the plastic dissipation term \( PD \) includes energy dissipated in both elastic plastic solids (soil) and contact/interface elements.

Figure 504.81 shows the accumulated plastic dissipation density field of the NPP model at the end of seismic event. The super-structure does not dissipate energy since it is modeled as a linear elastic material. Significant amount of seismic energy is dissipated in the contact/interface zone between the structure and underlying soil, especially at regions around the corners and edges of the foundation.
An arch-shaped elastic region is formed under the structure, where the soil moves together with the foundation and dissipates little energy. Such observation is consistent with classic bearing capacity analysis, which also indicates the formation of a relatively undeformed "active zone" beneath foundation.

As can be observed in Figure 504.82, the plastic dissipation density at location (A) is the highest. From Figure 504.72, it can be observed that more than 80% of the total input work is dissipated due to material plasticity or contact slipping. About 70% of the energy dissipation happens due to contact/interface slipping, which indicates that the property and behavior of the interface between foundation and soil is crucial in SSI system. It is worth pointing out that there is about 10% of the input work transformed into plastic free energy, which falls in the typical range reported by Taylor and Quinney (1934).

![Energy dissipation in SMR model for inelastic (elastic-plastic soil with contact).](image-url)
Figure 504.83: Animation of energy dissipation for an NPP.
504.6.5 Conclusion

Presented was a high fidelity seismic simulation methodology for investigating non-linear SSI effects on NPP structures. The site being in foot-wall of the reverse thrust fault, results in permanent subsidence in vertical direction. Due to plastification, elastic-plastic soil produces comparatively more vertical settlement than linear elastic soil. It also leads to damping of some higher frequency waves. This effect of non-linear material could be beneficial to the machines which are fatal to high frequency waves. It was found that with elastic-plastic soil, there was comparatively less seismic excitation and deformation in the NPP structure for the considered seismic motion. This illustrates that the stiff soil does not necessarily help in seismic behavior of structure. This also emphasizes the fact that linear elastic modeling of soil can lead to wrong conclusions resulting in huge capital loss. With the advancement of super-computers, uncertainty in modeling can be significantly reduced by following the high-fidelity modelling techniques discussed in this paper.

The non-linear effect studied here is with respect to the specific motion (Mw 5.5 up-dip slip fault). The non-linear effect cannot be fully described using this motion itself. More similar kind of research studies using different motions and geology needs to be carried out to find out the overall non-linear and geology effects on soil-structure interaction. The author also feels that new quantities needs to be formulated to study and compare different models to categories and unify the nonlinear effects on Soil Structure Interaction (SSI) effects. Energy dissipation analysis showed that the soil close to the corners and edges of the NPP structure dissipates large amount of seismic energy. An arch-shaped elastic region was identified where design can be improved so that soil strength at these locations can contribute to the overall safety of the SSI system.
504.7 3D Nonlinear Modeling for Small Modular Reactors (SMRs)

Figure 504.84 shows a generic model (half is only shown, full model is simulated) of an SMR. It is important to note extensive contact/interface zone of SMR walls with surrounding soil. This brings forward a number of modeling and simulation issues for SMRs:

- **Seismic Motions:** There will be a difference in seismic wave fields at the surface and at depth. Surface waves do extend somewhat into depth (about two wave lengths at most Aki and Richards (2002)), so different motion frequencies, wave lengths, and depending on soil stiffness, SMR will experience very different motions at the surface and at the base. This seismic motion incoherence will affect seismic response of an SMR.

- **Nonlinear/Inelastic Contact:** Large contact/interface zone, with its nonlinear/inelastic behavior will have significant effect on dynamic response of a deeply embedded SMR.

- **Nonlinear/Inelastic Soil Behavior:** With deep embeddement, dynamic behavior an SMR is significantly influenced by the nonlinear/inelastic behavior of adjacent soil.

- **Buoyant Forces:** With deep embeddement, and (a possible) presence of underground water (water table that is within depth of embeddement), water pressure on walls of SMR will create buoyant forces. During earthquake shaking, those forces will change dynamically, with possibility of cyclic mobility and liquefaction (even for dense soil, due to water pumping during shaking).

- **Uncertainty in Motions and Material:** Due to large contact/interface area and significant embeddement, significant uncertainty and variability (incoherence) in seismic motions will be present.
Moreover, uncertainties in properties of soil material surrounding SMR will add to uncertainty of the response.

Figure 504.85: Four main issues for realistic modeling of Earthquake Soil Structure Interaction of SMRs: variable weave field at depth and surface, inelastic behavior of contact/interface and adjacent soil, dynamic buoyant forces, and uncertain seismic motions and material.

Writeup that follows is based on Wang et al. (2017).

It noted that input files for this model are available at this LINK, and can be directly simulated using Real-ESSI Simulator, http://real-essi.info/, that is available on Amazon Web Services, https://aws.amazon.com/.

504.7.1 Introduction

Seismic performance of nuclear facilities is carefully analyzed considering the significant problems that damage of such structures can bring. The structure investigated here is a deeply embedded Small
Modular Reactor (SMR). Model SMR structure analyzed here is deeply embedded (36 meters) with only 14 meters of structure above ground.

In recent years, many researchers (Spyrakos et al. (1989); El Ganainy and El Naggar (2009); Iida (2012)) made efforts to perform realistic modeling of dynamic SSI system and seismic response of underground structure. Romero et al. (2013) coupled FEM and BEM method to model wave propagation in elastic foundation and corresponding dynamic response of the structure. Fatahi and Tabatabaiefar (2013) investigated the seismic performance of mid-rise buildings on soft soils using existing earthquake records. An elastoplastic SSI analysis was conducted by Shahrour et al. (2010) to explore the seismic response of tunnels in soft soils. However, some inherent modeling uncertainties still existed in these previous studies and were not well addressed:

- A very important modeling uncertainty comes from the ground motion. For surface structures, it is common to use historical earthquake records and simplified 1C seismic wave propagation models (Paolucci et al. (2008)). Vertical ground motion are usually neglected. However, Opsal and Fah (2007) has emphasized the necessity to use 3C ground motion by showing the big difference between 1C and 3C computation result. The modeling uncertainty of input motion for seismic modeling of underground structure is even higher. Due to the lack of ground motion observations along the depth, deconvolution method is usually adopted in many studies (Elgamal et al. (2008)) to get the excitation motion at certain depth. The deconvolution procedure represents a 1C linear inverse analysis. This inverse analysis is seemingly simple but it can introduce considerable confusion and uncertainties to the modeling system (Mejia and Dawson (2006)).

- Another uncertainty comes from the method that is used to input seismic motion into SSI system. Usually free field motion are directly imposed to the structure without considering ESSI effects. This is especially common for underground structures where simplified static loads are directly imposed and these structures are simply designed to accommodate the estimated free field deformation(Hashash et al. (2001)).

- Nonlinear ESSI effects are also important factor that is neglected or simplified in many existing studies. There are three sources of significant nonlinearity in an ESSI system: (a) Inelastic (elastic-plastic) behavior of soil, (b) inelastic (elastic-plastic) behavior of the contact/interface zone, and (c) inelastic (elastic-damage-plastic) behavior of the structure. Early works found that structural response can be quite different when elastoplasticity of surrounding soil is considered (Bielak (1978); Iguchi and Luco (1981)). In addition to that Jeremic et al. (2004) reported that ESSI behavior can have both beneficial and detrimental effects on structural behavior. The nonlinear
contact (interface) was analyzed by Hu and Pu (2004) and it was shown that its accurate modeling is a key part to realistic modeling of ESSI systems.

Due to computational limitations and complicated nature of ESSI problems mentioned above, there exist only few high-fidelity ESSI simulations, for bridges (Jeremić et al. (2009)) and tunnels (Corigliano et al. (2011)). To the Author’s knowledge, there is no high fidelity (realistic 3C motions, realistic elastic plastic soil modeling, and realistic contact/interface modeling) for a deeply embedded SMR structure. In this paper, we present high fidelity modeling of SMR using state-of-the-art ESSI methodology. Realistic 3C free field seismic motions are modeled using regional scale wave propagation models. Developed free field motions are then input into ESSI system using Domain Reduction Method (Bielak et al. (2003a)). Modeling description section presents inelastic/nonlinear modeling details for elastic-plastic models of surrounding soils and nonlinear interface/contact behavior. The inelastic/nonlinear modeling result are compared with linear elastic in Simulation Results section.

In addition to accurate modeling of 3C motions and nonlinear effects, energy propagation through the model is also accurately modeled. Energy dissipation is a widely used indicator of material damage in elastic plastic materials. A common misconception does exist, however, on the meaning of plastic work and plastic energy dissipation, as observed in a number of publications. Correct evaluation of energy dissipation should follow the principles of thermodynamics that incorporated plastic free energy (Rosakis et al., 2000; Dafalias et al., 2002). The thermodynamics framework presented by Yang et al. (2018, 2019) is implemented in the Real-ESSI Simulator Jeremić et al. (1988-2021), and is used to perform energy analysis on the SMR model in this paper. Energy dissipation in the SMR model is discussed in some detail.

504.7.2 Domain Reduction method

Input seismic motions into finite element model is an indispensable step for the simulation of soil structure interaction. The method we used here is called Domain Reduction Method, developed by Bielak et al. (2003a). It is a modular, two-step dynamic procedure aimed at reducing the large computation domain to a more manageable size. Firstly, large scale regional free field model is developed encompassing causative fault and location of SMR structure (however SMR structure is not present, it is a free field model). Time series of free field motions (displacements and accelerations) are recorded at locations of DRM elements, a single layer of finite elements encompassing soil structure SMR model. Those motions are then used in the second step, to develop effective forces that are used to input free field motions into ESSI SMR model.
504.7.3 3C Free Field Motions

Development of free field motions was done using a fourth order finite difference program SW4 (Petersson and Sjögreen (2018) developed at LLNL. Modeled was propagation of fault rupture in a model with dimensions $9 km \times 6 km \times 20 km$. The magnitude of simulated earthquake is 5.5. The shear wave velocity of soils in surface layer (500 meters thick) is $V_s = 500 m/s$. Motions were recorded in a box with dimensions $300 m \times 300 m \times 200 m$.

The characteristic ground motions recorded by ESSI nodes are plotted in Figure 504.86. The peak ground acceleration (PGA) in x and y direction is about 1g. Apart from that, significant amount of vertical motions with PGA 0.5g is also observed. The peak ground displacement (PGD) is about 0.1m in horizontal direction. Since ESSI box is located in the foot wall of the reverse fault, the permanent ground subsidence of about 6cm is recorded. Fourier transformation and response spectrum of the motions are shown in figure 504.87. The frequency range of the motion is within 15Hz. The dominant frequency of the motion is around 5 Hz. In response spectrum, we also see significant resonance effects for structure whose fundamental period is around 0.2s corresponding to 5 Hz fundamental frequency.
504.7.4 Model Description

In order to reduce model size using DRM method, we develop our target model with 6 layers. As shown in figure 504.88, the innermost part is a structure layer, which is surrounded by a soil layer. Following that, there is a DRM layer used to apply equivalent earthquake force. Outside DRM layer, there are three damping layers. These damping layers are designed to add high Rayleigh damping so that the outgoing wave (vibrations of structure, radiation damping) can be absorbed. Table 504.5 shows the material used. The size of whole FEM model is $72m \times 72m \times 56m$. There are 177,806 nodes, 20172 27-node brick elements, 3,177 contact/interface elements (modeling the interface between soil and embedded structure), with a total of over 533 thousand degrees of freedom (DoFs). The average mesh size is 3 meters. Newmark time integration method is used in this study with parameters $\gamma = 0.7$ and $\beta = 0.36$ adding numerical damping that reduces high frequency motions that were introduced by SW4. In order to capture the wave propagation in FEM model, mesh size should be controlled so that there is no artificial filtering to motions above certain frequency (Watanabe et al., 2017). As pointed out by Hughes (1987), 10 linear interpolation finite elements and 2 quadratic interpolation elements are needed per wave wavelength. Since second order 27 node brick element are used here, the minimum wave length captured is 6 meters. Considering shear wave velocity $v_s = 500m/s$, the maximum frequency calculated by equation 504.7 is 83 Hz. Even when material plasticifies (becomes softer), model is still propagating high frequencies of up to required $f_{max} \leq 15Hz$.

$$f_{max} = \frac{v_s}{\lambda_{min}}$$  \hspace{1cm} (504.7)
504.7.4.1 Embedded Nuclear Structure

Small Modular Reactor (SMR) analyzed here is a 4 storied reinforced concrete structure with total height 50 meters and 36 meters embedded in the ground. The length and width of the structure is 30 meters. The whole structure is modeled using 27-node solid brick element with linear elastic material. The Young’s modulus is selected as $E = 30 \text{GPa}$ and Poisson’s ratio $\nu = 0.2$. Single layer of 27 node bricks can accurately model (beam and plate) bending, and is hence chosen for structural model.

504.7.4.2 Soil Model

The depth of the model surrounding the structure is 45m. The soil is assumed to be saturated soil with undrained behavior during the earthquake. In order to considering nonlinear site effects, the soil is modeled with elastoplastic material. In the past 20 years, many 3D constitutive models Yang et al. (2003); Dafalias and Manzari (2004b); Park and Byrne (2004); Pisanò and Jeremić (2014) have been developed. Undrained behavior can be approximately modeled using von Mises material model (Yang and Jeremić, 2003). Elastic plastic von Mises material model with linear kinematic hardening rule is used here. The material parameters are presented in table 504.5. Backward Euler implicit algorithm (Jeremić and Sture (1997)) is used for the equilibrium iterations at constitutive level.

504.7.4.3 Soft Contact Element

Model for contact/interface (axial contact, gap opening and closing and slip behavior) of the interface between structure and surrounding soil, relies on a node-to-node soft contact/interface element (Sinha and Jeremić (2017)). In soft contact, the normal stiffness exponentially grows as the relative displacement between two contact/interface nodes increases and finally reaches maximum normal stiffness. 3,177 contact elements are placed at the soil-structure interface. Contact/Interface parameters are shown in Table 504.5.

504.7.4.4 Simulation Procedure

The nonlinear ESSI analysis was conducted using Real-ESSI Simulator (Jeremić et al. (1988-2021)) developed at UC Davis and LBNL. Two SMR simulation models were simulated. First model uses linear elastic soil without contact element and second model uses inelastic/nonlinear soil with inelastic/nonlinear contact. In both cases, two loading stages were modeled: First loading stage is a self weight, developed by adding a uniform gravity field. This is a necessary stage for inelastic analysis in order to develop initial stress state of structure and surrounding soil before earthquake comes. Self weight was also applied to the elastic model, just so that we have comparable displacement results. Then second loading stage
is an earthquake load. Simulations were performed on a local parallel computer with parallel version of the Real-ESSI Simulator using 10 CPUs. It is noted that Real-ESSI Simulator is also available on Amazon and Google cloud parallel computers as well as on large national parallel compters at the LBNLf (EDISON and CORI). Local parallel computer (a workstation) was used in order to illustrate versatility of Real-ESSI Simulator and to show that high fidelity parallel computations do not need to require high price parallel computers.

### 504.7.5 Simulation Results

Figure 504.89 shows time series acceleration response of top center of SMR.

The elastic results represent simulation case where the surrounding soil is modeled using linear elastic material and no contact elements in soil-structure interface. The inelastic results represent simulation case where the surrounding soil is modeled using inelastic soil material and an inelastic contact. Signifi-

<table>
<thead>
<tr>
<th>Table 504.5: Modeling parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material parameters</td>
</tr>
<tr>
<td>shear wave velocity [m/s]</td>
</tr>
<tr>
<td>Young’s modulus [GPa]</td>
</tr>
<tr>
<td>Poisson ratio</td>
</tr>
<tr>
<td>von Mises radius [kPa]</td>
</tr>
<tr>
<td>kinematic hardening rate [MPa]</td>
</tr>
<tr>
<td>Contact parameters</td>
</tr>
<tr>
<td>initial normal stiffness [N/m]</td>
</tr>
<tr>
<td>hardening rate [/m]</td>
</tr>
<tr>
<td>maximum normal stiffness [N/m]</td>
</tr>
<tr>
<td>tangential stiffness [N/m]</td>
</tr>
<tr>
<td>normal damping [N/(m/s)]</td>
</tr>
<tr>
<td>tangential damping [N/(m/s)]</td>
</tr>
<tr>
<td>friction ratio</td>
</tr>
<tr>
<td>Damping parameters</td>
</tr>
<tr>
<td>structure layer</td>
</tr>
<tr>
<td>surrounding soil</td>
</tr>
<tr>
<td>DRM layer</td>
</tr>
<tr>
<td>outside layer 1</td>
</tr>
<tr>
<td>outside layer 2</td>
</tr>
<tr>
<td>outside layer 3</td>
</tr>
</tbody>
</table>

...
cant acceleration decreases can be seen in the inelastic case. The horizontal peak acceleration values is reduced by almost 40%. This is due to plastification of soil and inelastic response of the contact/interface zone. Moreover significant seismic energy is dissipated, so that inelastic soil and inelastic contact/interface act as seismic dissipators and isolators during seismic event. The acceleration difference in vertical, z direction is less significant than horizontal direction.

Figure 504.90 shows Fourier magnitude that high frequency component of horizontal acceleration was significant decreased in inelastic case.

Material Modeling Influence on Results is Shown in Figure 504.91.
SMR: Variation of Acceleration with Depth is Shown in Figures 504.92 and 504.93.
SMR: PGA and PGD Depth Variation is Shown in Figure 504.94.

- The PGA & PGD of SSI systems are (very) different from free field motions,
- Material nonlinearity has significant effect on acceleration response.

SMR: Elastic vs Inelastic response is shown in Figure 504.95.

Figure 504.96 shows the distribution of plastic strain in surrounding soil. There are two main plastic zones near two bottom corners of the structure. Also the plastic strain at the soil-structure interface is higher than adjacent area. It is interesting to note that there is an elastic zone beneath structure. The
Figure 504.91: Upper: Material A: nonlinear, vM - AF; Lower: Material B: Bilinear.

Figure 504.92: Free field response, left: elastic, right, inelastic.

Figure 504.93: ESSI for an SMR response, left: elastic, right, inelastic.
shape of the elastic zone resembles a bulb and is due to self weight and stiffness of the structure that compresses soil beneath.
Figure 504.96: Distribution of the magnitude of plastic strain
504.7.6 Energy Dissipation

According to the thermodynamics framework presented by Yang et al. (2019), the energy dissipation in any decoupled material undergoing isothermal process can be expressed as:

$$\Phi = \sigma_{ij} \dot{\epsilon}_{ij} - \sigma_{ij} \dot{\epsilon}_{ij}^{el} - \rho \dot{\psi}_{pl} \geq 0$$  \hspace{1cm} (504.8)

where $\Phi$ is the rate of change of energy dissipation per unit volume (or dissipation density), $\sigma_{ij}$ and $\epsilon_{ij}$ are the stress and strain tensors respectively, $\epsilon_{ij}^{el}$ is the elastic part of the strain tensor, $\rho$ is the mass density of the material, and $\psi_{pl}$ is the plastic free energy per unit volume (or plastic free energy density). Equation 504.8 ensures the energy balance and nonnegative energy dissipation conditions that correspond to the first and second law of thermodynamics.

With Equation 504.8, the energy balance of a SSI system is simply given by:

$$W_{Input} = E_{Stored} + E_{Dissipated} = KE + SE + PF + PD$$  \hspace{1cm} (504.9)

where $W_{Input}$ is the input work due to external loading, $KE$ is the kinetic energy, $SE$ is the elastic strain energy, $PF$ is the plastic free energy, and $PD$ is the energy dissipation due to material plasticity. Formulation for each energy component can be found in Yang et al. (2018). Note that in Equation 504.9, it is assumed that no other forms of energy dissipation exists in the system.

Figure 504.97 (a) shows the distribution of plastic dissipation density in the SMR model at the end of simulation. The case presented in this section is elastic plastic soil without contact/interface element. Note that the structure is modeled with elastic material, so they do not dissipate any energy. As expected, more seismic energy is dissipated around the corners and edges of the structure due to stress concentration. It can be observed that there are several elastic regions around the boundaries of the structure, which means that the soil there does not plastify much and moves together with the structure. Economy of the design can be improved by better utilizing the strength of soil around these locations.

Figure 504.97 (b) show the evolution of energy components at location A. It can be observed that the amount of plastic energy dissipation is much larger than the other forms of energy, indicating that the nonlinear effect is quite significant in deeply embedded structure. Another interesting observation is the small amount of plastic free energy whose quantity largely depends on material hardening parameters and loading conditions. It should be pointed out that even if it is small, plastic free energy should never be neglected so that the condition of nonnegative incremental energy dissipation can be upheld Rosakis et al. (2000); Taylor and Quinney (1934).

SMR: Energy Dissipation for an SMR, Figure 504.98.
Figure 504.97: Energy dissipation in SMR model: (a) Plastic dissipation density field at the end of simulation; (b) Evolution of energy components at location A.
Figure 504.98: Animation of energy dissipation for an SMR.
504.7.7 Conclusion

The seismic response of an embedded SMR has been modeled with high fidelity. Using state-of-the-art nonlinear ESSI simulation techniques, many modeling uncertainties have been eliminated. The methodology shown here is also applicable to many other ESSI problems (buildings, bridges, dams, etc.). The simulation result of SMR shows that the acceleration response of the structure decreases with nonlinear effects properly modeled. In addition, the high frequency component of acceleration is significantly damped out in inelastic case due to soil plastification.

Energy dissipation analysis shows that the soil close to the edge of the SMR structure dissipates large amount of seismic energy during shaking. Such observation also indicates significant nonlinear effect when elastoplastic material is used for soil modeling. Several elastic regions are identified where design can be improved so that soil strength at these locations can contribute to the safety of the SSI system.
504.8 Three Dimensional (3D) Inelastic Modeling for Structure Soil Structure Interaction
Chapter 505

Liquefaction and Cyclic Mobility

(2002-2006-2009-2021-)

(In collaboration with Dr. Zhao Cheng, and Dr. Panagiota Tasiopoulou)
505.1 Chapter Summary and Highlights

505.2 Introduction

Liquefaction is one of the most complex phenomena in earthquake engineering. Liquefaction also represents one of the biggest contributors to damage of constructed facilities during earthquakes (Kramer, 1996a). Prediction of behaviors of liquefiable soils is difficult but achievable. There are number of methods that can be utilized to predict such behaviors. Methods currently used can have varying prediction accuracy and certainty. Of particular interest in this paper is the description of verified and validated numerical simulation methodology based on rational mechanics that is used to model, simulate and predict behavior of a single pile in liquefiable soil subjected to seismic loading. Both level and sloping ground pile systems are modeled and simulated. Detailed description of background theory, formulation and implementation were recently given by Cheng et al. (2007) and Jeremić et al. (2008).

It should be noted that presented development does show great promise in analyzing a myriad of liquefaction related problems in geotechnical and structural engineering. The effectiveness and power of numerical simulation tools for analyzing liquefaction problems becomes even more important and prominent in the light of potential disadvantages of models used in experimental simulations. These disadvantages, related to proper scaling (Wood, 2004) and problems in maintaining appropriate similarities (Harris and Sabnis, 1999) for first order important phenomena, can render scaled models ineffective, when used for physical simulations (under one-step or multiple-step gravity loading).

In what follows, a brief literature review is provided. The literature review comprises sections on observations of liquefaction behavior in case studies, non–continuum modeling efforts, review of redistribution of voids and pore fluid volume/pressures phenomena and continuum modeling efforts.

Observation of Behavior. Liquefaction behavior was observed during a number of earthquakes in the past. During Alaskan Earthquake (1964), liquefaction was the main cause of severe damage to 92 highway bridges, moderate to light damage to another 49 highway bridges, and moderate to sever damage to 75 railroad bridges (Youd and Bartlett, 1989). During Niigata Earthquake (1964) liquefaction induced damage to foundation piles under Yachiya bridge (Hamada, 1992). During that same earthquake, girders of Showa Bridge toppled as the support structure and piles moved excessively due to liquefaction (Japanese Society of Civil Engineers, 1966). During Kobe Earthquake (1995), liquefaction was the primary cause of damage to many pile supported or caisson supported bridges and structures. For example, Shin–Shukugawa bridge was subjected to excessive pile foundation movement due to liquefaction (Yokoyama et al., 1997).

Opposed to these failures and collapses, there were a number of bridges with pile foundations that
did not suffer much or even minor damage even though there was liquefaction around foundations. For example, pile foundations of the Landing Road Bridge in New Zealand performed quite well during Edgecumbe earthquake (1987) even with a significant liquefaction recorded (Berril et al., 1997; Dobry and Abdoun, 2001). In addition to that, Second Maya Bridge piles (large steel pipes) were not damaged during Kobe earthquake despite significant liquefaction in surrounding soils (Yokoyama et al., 1997).

**Non–Continuum Modeling Efforts.** Modeling and simulation of piles in liquefied grounds has been focus of a number of recent studies. The simple approach, based on scaling of p-y springs has been suggested early by Japanese Road Association (1980), Architectural Institutive of Japan (1988), Liu and Dobry (1995), Miura et al. (1989) and O’Rourke (1991). However, large inconsistencies with material parameter selection are present when p-y spring approach is used for piles in liquefied soils. Since p-y methodology for liquefied soils is not based on rational mechanics, appropriate choice of material parameters is primarily based on empirical observations of behaviors of piles in liquefied soils in experimental studies. A number of experimental studies have carefully examined pile behaviors in liquefiable soils. We mention Tokida et al. (1992), Liu and Dobry (1995), Abdoun et al. (1997), Horikoshi et al. (1998) and Boulanger and Tokimatsu (2006). Studies using physical model can be used to obtain very high quality data on behavior of piles in liquefied soils, provided that similarity of important physical phenomena is maintained (Wood, 2004; Harris and Sabnis, 1999). Some of the recent papers that discussed use of these models and gave recommendations about parameter choices are listed for reference: Tokimatsu and Asaka (1998), Martin et al. (2002), Dobry et al. (2003), Liyanapathirana and Poulos (2005), Rollins et al. (2005), Čubrinovski and Ishihara (2006), Brandenberg et al. (2007).

**Redistribution of Voids and Pore Fluid Volume/Pressures.** Mechanics of pile behavior in liquefiable grounds is based on the concept of redistribution of voids and pore fluid volume/pressures (RVPFVP). It should be emphasized that geomechanics phenomena of redistribution of voids – pore fluid volume/pressure is used here in purely mechanistic way. That is, RVPFVP is a phenomena that occurs in saturated soils and that phenomena is responsible for (is manifested in) liquefaction related soil behaviors with or without piles. This is noted as in some recent publications, RVPFVP terminology is explicitly used for problems of liquefaction induced failures of sloping grounds without piles. Our understanding of the RVPFVP phenomena is that RVFVP is responsible for many more facets of behavior of liquefied soils, rather than only failure of liquefied slopes.

The early investigation of the RVPFVP phenomena was related to the behavior of infinite slopes. For example, loss of shear strength in infinite slopes is one of the early understood manifestations of RVPFVP (Whitman, 1985; National Research Council, 1985; Malvick et al., 2006). Laboratory investigation of
sand was also used to observe the RVPFVP phenomena (Casagrande and Rendon, 1978; Gilbert, 1984).

**Continuum Modeling Efforts.** Continuum based formulations for modeling liquefaction problems have been present for over two decades. In a landmark paper, Zienkiewicz and Shiomi (1984) presented three possible coupled formulations for modeling of soil skeleton – pore fluid problems. The most general and complete one is the so called u-p-U formulations while the other two, the u-p and the u-U have a number of restrictions on the domain of application. Here, the unknowns are the soil skeleton displacements $u$; the pore fluid (water) pressure $p$; and the pore fluid (water) displacements $U$. The u-p formulation captures the movements of the soil skeleton and the change of the pore pressure, and is the most simplistic one of the three mentioned above. This formulation neglects the differential accelerations of the pore fluid (it does account for acceleration of pore fluid together with soil skeleton, but not the differential one if it exists), and in one version neglects the compressibility of the fluid (assuming complete incompressibility of the pore fluid). In the case of incompressible pore fluid, the formulation requires special treatment of the approximation function (shape function) for pore fluid to prevent the volumetric locking (Zienkiewicz and Taylor, 2000). The majority of the currently available implementations are based on this formulation. For example Elgamal et al. (2002) and Elgamal et al. (2003) developed an implementation of the u-p formulation with the multi-surface plasticity model by Prevost (1985b), while Chan (1988) and Zienkiewicz et al. (1999a) used generalized theory of plasticity Pastor et al. (1990).

The u-U formulations tracks the movements of both the soil skeleton and the pore fluid. This formulation is complete in the sense of basic variables, but might still experience numerical problems (volumetric locking) if the difference in volumetric compressibility of the pore fluid and the solid skeleton is large.

The u-p-U formulation resolves the issues of volumetric locking by including the displacements of both the solid skeleton and the pore fluid, and the pore fluid pressure as well. This formulation uses additional dependent unknown field of pore fluid pressures to stabilize the solution of the coupled system. The pore fluid pressures are connected to (dependent on) displacements of pore fluid. With known (given) volumetric compressibility of the pore fluid, pore fluid pressure can be calculated. Despite it’s power, the u-p-U formulation has rarely been implemented into finite element code, and has never (at least to our knowledge) been used to analyze pile – liquefied soil interaction. This can be attributed in part to a sophistication of implementation that is required, and to a sizable increase in computational cost for u-p-U elements.
505.3 Liquefaction of Level and Sloping Grounds

Material presented here is from Cheng et al. (2007); Jeremić et al. (2008).

Liquefaction of level and sloping grounds represents a very common behavior during earthquakes. Of interest is to estimate settlement for level ground, and horizontal movements for sloping grounds. In next few sections, presented are results for a 1D, vertical (level ground) and sloping ground cases for dense and loose sand behavior during seismic shaking.

505.3.1 Model Description

Vertical soil column consists of a multiple-elements subjected to an earthquake shaking. The soil is assumed to be Toyoura sand and the calibrated parameters are from Dafalias and Manzari (2004a), and are given in the Table (505.1).

Table 505.1: Material parameters of Dafalias-Manzari model.

<table>
<thead>
<tr>
<th>material parameter</th>
<th>value</th>
<th>material parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elasticity</td>
<td>$G_0$</td>
<td>125 kPa</td>
<td>$h_0$</td>
</tr>
<tr>
<td></td>
<td>$v$</td>
<td>0.05</td>
<td>$c_h$</td>
</tr>
<tr>
<td>Critical state</td>
<td>$M$</td>
<td>1.25</td>
<td>$n_b$</td>
</tr>
<tr>
<td></td>
<td>$c$</td>
<td>0.712</td>
<td>$A_0$</td>
</tr>
<tr>
<td></td>
<td>$\lambda_c$</td>
<td>0.019</td>
<td>$n_d$</td>
</tr>
<tr>
<td></td>
<td>$\xi$</td>
<td>0.7</td>
<td>$z_{max}$</td>
</tr>
<tr>
<td></td>
<td>$e_r$</td>
<td>0.934</td>
<td>$c_z$</td>
</tr>
<tr>
<td>Yield surface</td>
<td>$m$</td>
<td>0.01</td>
<td></td>
</tr>
</tbody>
</table>

The other parameters, related to the boundary value problem are given in table (505.2).

For tracking convenience, the mesh elements are labeled from E01 (bottom) to E10 (surface) and nodes at each layers are labeled from A (bottom) to K (surface).

A static application of gravity analysis is performed before seismic excitation. The resulting fluid hydrostatic pressures and soil stress states along the soil column serve as initial conditions for the subsequent dynamic analysis.

It should be noted that the self weight loading is performed on an initially zero stress (unloaded) soil column and that the material model and numerical integration algorithms are powerful enough to follow through this early loading with proper evolution. The boundary conditions are such that the soil and water displacement degree of freedom (DOF) at the bottom surface are fixed, while the pore pressure
Table 505.2: Additional parameters used in boundary value problem simulations (other than material parameters from the Table (505.1)).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid density</td>
<td>$\rho_s$</td>
<td>2700 kg/m$^3$</td>
</tr>
<tr>
<td>Fluid density</td>
<td>$\rho_f$</td>
<td>1000 kg/m$^3$</td>
</tr>
<tr>
<td>Solid particle bulk modulus</td>
<td>$K_s$</td>
<td>$3.6 \times 10^7$ kN/m$^2$</td>
</tr>
<tr>
<td>Fluid bulk modulus</td>
<td>$K_f$</td>
<td>$2.2 \times 10^6$ kN/m$^2$</td>
</tr>
<tr>
<td>permeability</td>
<td>$k$</td>
<td>$5.0 \times 10^{-4}$ m/s</td>
</tr>
<tr>
<td>HHT parameter</td>
<td>$\alpha$</td>
<td>-0.2</td>
</tr>
</tbody>
</table>

DOFs are free; the soil and water displacement DOFs at the upper surface are free upwards to simulate the upward drainage. The pore pressure DOFs are fixed at surface thus setting pore pressure to zero. On the sides, soil skeleton and water are prevented from moving in horizontal directions while vertical movement of both is free. It is emphasized that those displacements (of soil skeleton and pore fluid) are different. In order to simulate the 1D behavior, all DOFs at the same depth level are connected in a master–slave fashion. Modeling of sloping ground is done by creating a constant horizontal load, sine of inclination angle, multiplied by the self weight of soil column, to mimic sloping ground. In addition to that, for a sloping ground, there should be a constant flow (slow) downhill, however this is neglected in our modeling. The permeability is assumed to be isotropic $k = 5.0 \times 10^{-4}$ m/s. The input acceleration time history (Figure (505.1)) is taken from the recorded horizontal acceleration of Model No.1 of VELACS project Arulanandan and Scott (1993) by Rensselaer Polytechnic Institute, http://geoinfo.usc.edu/gees/velacs/. The magnitude of the motion is close to 0.2 g, while main shaking lasts for about 12 seconds (from 1 s to 13 s). For the sloping ground model a slope of % 3 was considered.

It should be emphasized that the soil parameters are related to Toyoura sand, not Nevada sand which is used in VELACS project. The purpose of presented simulation is to show the predictive performance using verified and validated formulation, algorithms, implementation and models.

505.3.2 Behavior of Saturated Level Ground

Figure (505.2) describes the response of the sample with loose sand $e_0 = 0.85$. This figure shows the typical mechanism of cyclic decrease in effective vertical stress due to pore pressure build up as expected for the looser than critical granular material. The lower layers show only the reduction of effective vertical stress from the beginning. Once the effective vertical (and therefore confining) stress...
approaches the smaller values, signs of the so-called butterfly shape can be observed in the stress path. Similar observation can be made in the upper layers which have the smaller confining pressure comparing to the lower layers from the very beginning due to lower surcharge. The upper layers have lower confining pressure (lower surcharge) at the beginning of the shaking, hence less contractive response is expected in these layers; however, soon after the initiation of the shaking these top layers start showing the liquefaction state and that type of response continues even after the end of the shaking. The top section of the model has remained liquefied well past the end of shaking. This is explained by the large supply of pore fluid from lower layers, for which the dissipation starts earlier. For example, for the lowest layer, the observable drop in excess pore pressure starts as soon as the shaking ends, while, the upper layers then receive this dissipated pore fluid from lower layers and do liquefy (or continue being liquefied) well past end of shaking (which happens at approximately 13 seconds). It is very important to note the significance of this incoming pore water flux on the pore water pressure of the top layers. Despite the less contractive response of soil skeleton at the top elements, the transient pore water flux, that enters these elements from the bottom, forces those to a liquefaction state. In other words, the top elements have not liquefied only due to their loose state but also because of the water flow coming from the bottom layers. The maximum horizontal strains can be observed in the middle layers due to liquefaction and prevents upper layers from experiencing larger strains. The displacements of water and soil are presented in the last column. It shows that in all layers the upward displacement of water is larger than the downward displacement of soil. This behavior reflects soil densification during shaking.

Figure 505.1: Input earthquake ground motion for the soil column.
Figure (505.3) describes the response of the sample with dense sand \((e_0 = 0.75)\). This figure also shows the typical mechanism of cyclic decrease in effective vertical stress. However, in case of this dense sample the decreasing rate of the effective confining pressure is much smaller than what was observed in the loose sample. Signs of the partial butterfly shape in the effective stress path can be observed from early stages of shaking. The butterfly is more evident in the upper layers with the lower confining pressure, i.e. more dilative response. In later stages of the shaking, i.e. when the confining pressure reduces to smaller value the butterfly shape of the stress path gets more pronounced due to having more dilative response in the lower confining pressure based on CSSM concept. In comparing this dense case to the case of shaking the loose sand column, the current case does not show any major sign of liquefaction (when stress ratio \(r_u = 1\)). This is due to the less contractive (more dilative) response of the sand in this case, which is coming from the the denser state of the sample. Because of having partial segments of dilative response, the whole column of the sand has not loosed its strength to the extent that happened for the case of loose sand and therefore smaller values of horizontal strains has been observed in the results. The absolute values of soil and water vertical displacements are also smaller than the case of loose sand which can be again referred to the less overall contractive response in this case.

Overall, it can be noted that the response in the case of loose sand \((e_0 = 0.85)\) is mainly below the dilatancy surface (phase transformation surface) while the denser sand sample with \(e_0 = 0.75\) shows partially dilative response referring to the denser than critical state.

### 505.3.3 Behavior of Saturated Sloping Ground

Figures (505.4) and (505.5) present the result of the numerical simulations for shaking the inclined soil columns (toward right) with loose and dense sand samples, respectively. The inclination of the soil column results in presence of the offset shear stress to the right side. This essentially poses asymmetric horizontal shear stresses (toward the direction of inclination) during cycles of shaking. On one hand, this offset shear stress makes the sample more dilative in the parts of shaking toward the right side (think about the state distance from the phase transformation line or dilatancy line in the \(p - q\) space). As a result asymmetric butterfly loops will be induced causing the soil to regain its stiffness and strength \((p)\) in the dilative parts of the corresponding cycles, therefore only instantaneous spikes of \(r_u = 1\) can be observed in case of the sloped columns of soil. There is also a permanent liquefaction in terms of having stationary portions of \(r_u = 1\) in this case. On the other hand, the offset shear stress results generation of more horizontal strains in the portions of loading which are directed toward the right side than those which are directed back toward the left side. As a result the horizontal shear strains will accumulative toward the right side and create larger permanent horizontal displacement comparing...
Figure 505.2: Seismic results for (loose sand) soil column in level ground \((e_0 = 0.85)\).

Figure 505.3: Seismic results for (dense sand) soil column in level ground \((e_0 = 0.75)\).
to the case of level ground soil column. Since the overall dilative response of the dense sample, i.e. Figure (505.5), is larger than that of the loose sample, i.e. Figure (505.4), the dense sample shows stiffer response and therefore less accumulative horizontal shear strains than the loose sample. The difference in predicted horizontal displacements is almost three times, that is, for the dense sample the final, maximum horizontal displacement is approx. 0.5 m, while for the loose sand sample, it almost 1.5 m.

![Figure 505.4: Seismic results for (loose sand) soil column in sloping ground ($\varepsilon_0 = 0.85$).](image)

### 505.4 Pile in Liquefied Ground, Staged Simulation Model Development

Material presented here is from Cheng and Jeremić (2009a,b) Model development for a pile in the liquefiable soil follows physics (mechanics) of the problem as close as possible. Numerical simulation of such problems in geomechanics is usually based on stages of loading and increments within those stages.

All load stages are applied to a series of finite element models, all of which share features of an initial soil model. This initial soil model consists of a soil block with dimension of $12 \times 12 \times 15$ m (length $\times$ width $\times$ depth). Due to the symmetry of the model, only half of the block is modeled. Symmetry assumptions is based on assumption that all the loads, dynamic shaking and other influences are symmetric with
respect to the plane of symmetry. This specialization to symmetric model reduces model generality (for example this use of symmetry will preclude analysis of dynamic shaking perpendicular to sloping ground dip). However, as our goal is to present a methodology of analyzing behavior of piles in liquefying ground, this potential drawback is not deemed significant in this study. Finite element mesh for the model is presented in Figure (505.6). The initial mesh consists of 160 eight node u-p-U elements.

Each node of the mesh has 7 degrees of freedom, three for soil skeleton displacements \( u_i \), one for pore water pressure \( p \), and three for pore water displacement \( U_i \). While it can be argued that the mesh is somewhat coarse, it is well refined around the pile, yet to be installed, in place of gray region in the middle.

A single set of parameters is used with the Dafalias-Manzari material model. Soil is modeled as Toyoura sand and material parameters (summarized in Table 505.4) are calibrated using tests by Verdugo and Ishihara (1996), while initial void ration was set to \( e_0 = 0.80 \). It is very important to emphasize that the state of stress and internal variables from initial state (zero for stress and given value for void ratio and fabric) will evolve through all stages of loading by proper modeling and algorithms, by using single set of material parameters. Table 505.4 presents additional parameters, other than material parameters

Figure 505.5: Seismic results for (dense sand) soil column in sloping ground \( (e_0 = 0.75) \).
Figure 505.6: Left: Three dimensional finite element mesh featuring initial soil setup, where all the soil elements are present. The gray region of elements is excavated (numerically) and replaced by a pile during later stages of loading; Right: Side view of the pile-soil model with some element and node annotation, used to visualize results.

presented in Table 505.4, used for numerical simulations.

505.4.1 First Loading Stage: Self Weight

The initial stage of loading is represented by the application of self weight on soil, including both the soil skeleton and the pore water. Initial state in soil before application of self weight is of a zero stress and strain while void ratio and fabric are given initial values. The state of stress/strain, void ratio and fabric will evolve upon application of self weight. At the end of self weight loading stage, soil is under appropriate state of stress ($K_0$ stress), the void ratio corresponds to the void ratio after self weight (redistributed such that soil is denser at lower layers), while soil fabric has evolved with respect to stress.
Table 505.3: Material parameters used for Dafalias-Manzari elastic-plastic model.

<table>
<thead>
<tr>
<th>Material Parameter</th>
<th>Value</th>
<th>Material Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elasticity</td>
<td>$G_0$</td>
<td>125 kPa</td>
<td>$h_0$</td>
</tr>
<tr>
<td>v</td>
<td>0.05</td>
<td></td>
<td>$c_h$</td>
</tr>
<tr>
<td>Critical state</td>
<td>$M$</td>
<td>1.25</td>
<td>$n_b$</td>
</tr>
<tr>
<td>$c$</td>
<td>0.8</td>
<td>Dilatancy</td>
<td>$A_0$</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>0.019</td>
<td></td>
<td>$n_d$</td>
</tr>
<tr>
<td>$\xi$</td>
<td>0.7</td>
<td>Fabric-dilatancy</td>
<td>$\zeta_{max}$</td>
</tr>
<tr>
<td>$e_r$</td>
<td>0.934</td>
<td></td>
<td>$c_z$</td>
</tr>
<tr>
<td>Yield surface</td>
<td>$m$</td>
<td>0.02</td>
<td></td>
</tr>
</tbody>
</table>

Table 505.4: Additional parameters used in FEM simulations.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid density</td>
<td>$\rho_s$</td>
</tr>
<tr>
<td>Fluid density</td>
<td>$\rho_f$</td>
</tr>
<tr>
<td>Solid particle bulk modulus</td>
<td>$K_s$</td>
</tr>
<tr>
<td>Pore fluid bulk modulus</td>
<td>$K_f$</td>
</tr>
<tr>
<td>permeability</td>
<td>$k$</td>
</tr>
<tr>
<td>Gravity</td>
<td>$g$</td>
</tr>
</tbody>
</table>

induced anisotropy. All of these changes are modeled using Dafalias–Manzari material model and using constitutive and finite element level integration algorithms developed within UC Davis Computational Geomechanics group in recent years.

Boundary conditions (BC) for self weight stage of loading are set in the following way:

- Soil skeleton displacements ($u_i$), are fixed in all three directions at the bottom of the model. At the side planes, nodes move only vertically to mimic self-weight effect. All other nodes are free to move in any direction.

- Pore water pressures ($p$), are free to develop at the bottom plane and at all levels of the models except at the top level at soil surface where they are fixed (set to zero, replicating drained condition).

- Pore water displacements ($U_i$), are fixed in all three directions at the bottom, are free to move only vertically at four sides of the model and are free to move in any direction at all other nodes.
These boundary conditions are consistent with initial self-weighting deformation conditions for soil and pore water at the site.

For the case of sloping ground, an additional load sub-stage is applied after self weight loading, in order to mimic self weight of inclined (sloping) ground. This is effectively achieved by applying a resultant of total self weight of the soil skeleton times the sine of the inclination angle at uphill side of the model. This load is applied only to the solid skeleton DOFs, and not on the water DOFs. Physically it would be correct to consider the sloping ground effects on the pore water as well. This will create a constant flow field of the water downstream, which, while physically accurate, is small enough that it does not have any real effect on modeling and simulations performed here.

### 505.4.2 Second Loading Stage: Pile–Column Installation

After the first loading stage, comprising self weight applications (for level or sloping ground, as discussed above), second loading stage includes installation (construction) of the pile–column. Modeling changes performed during loading stage included:

- Excavation of soil occupying space where the pile will be installed. This was done by removing elements, nodes and loads on elements shown in gray in Figure (505.6).

- These elements were replaced by very soft set of elements with small stiffness, low permeability. This was done in order to prevent water from rushing into the newly opened hole in the ground after original soil elements (used in the first loading stage) are removed.

- Installation of a pile in the ground and a superstructure (column) above the ground. Nonlinear beam–column elements were used for both pile and column together with addition of appropriate nodal masses at each beam-column node, and with the addition of a larger mass at the top representing lumped mass of a bridge superstructure. Pile beam-column elements were connected with soil skeleton part of soil elements using a specially devised technique.

As mentioned earlier, the volume that would be physically occupied by the pile in the pile hole, is “excavated” during this loading stage. Beam–column elements, representing piles, are then placed in the middle of this opening. Pile (beam–column) elements are then connected to the surrounding soil elements by means of stiff elastic beam–column elements. These “connection” beam–column elements extend from each pile node to surrounding nodes of soil elements. The connectivity of nodes to soil skeleton nodes is done only for three beam–column translational DOFs, while the three rotational DOFs from the beam–column element are left unconnected. These three DOFs from the beam–column side are connected to first three DOFs of the u-p-U soil elements, representing displacements of the soil skeleton
(\(u_i\)). Water displacements (\(U_i\)) and pore water pressures (\(p\)) are not connected in any way. Rather, these two sets of DOFs representing pore water behave in a physical manner (cannot enter newly created hole around pile beam–column elements) because of the addition of a soft, but very impermeable set of \(u\)-\(p\)-\(U\) elements, replacing excavated soil elements. By using this method, both solid phase (pile with soil skeleton) and the water phase (pore water within the soil) are appropriately modeled. Figure (505.7) shows in some detail schematics of coupling between the pile and soil skeleton part finite elements.

![Schematic description of coupling of displacement DOFs (\(u_i\)) of beam-column element (pile) with displacement DOFs (\(u_i\)) of \(u\)-\(p\)-\(U\) elements (soil).](image)

Figure 505.7: Schematic description of coupling of displacement DOFs (\(u_i\)) of beam-column element (pile) with displacement DOFs (\(u_i\)) of \(u\)-\(p\)-\(U\) elements (soil).

Nonlinear force based beam–column elements (Spacone et al., 1996a,b) were used for modeling the pile–column. Pile was assumed to be made of aluminum. This was done in order to be able to validate simulations with centrifuge experiments (when they become available). Presented models were all done in prototype scale, while for (possible future) validation, select results will be carefully scaled and compared with appropriate centrifuge modeling. Pile and the column were assumed to have a diameter of \(d = 1.0\) m, with Young’s modulus of \(E = 68.5\) GPa, yield strength \(f_y = 255\) kPa, and the density \(\rho = 2.7\) kg/m³. Wall thickness of prototype pile–column is \(t = 0.05\) m. Lumped mass of pile and column was distributed along the beam–column nodes, while an additional mass was added on top \(m = 1200\) kg that represents (small) part of the superstructure mass. This particular mass \(m = 1200\) kg comes from a standard (scaled up in our case) centrifuge model for pile–column–mass used at UCD.

Figure (505.6) (right side) shows side view of the column-pile-soil model after second stage of loading.
### 505.4.3 Third Loading Stage: Seismic Shaking

After the application of self weight on the uniform soil profile, excavation and construction of the single pile with column and super structure mass on top and application of their self weight, the model is at the appropriate initial state for further application of loading. In this case, this additional loading comprises seismic shaking. For this stage, fixed horizontal DOFs used on the side planes during the first stage are removed (set free).

The input acceleration time history, shown in Figure (505.8) was taken from the recorded horizontal acceleration of Model No.1 of VELACS project Arulanandan and Scott (1993) by Rensselaer Polytechnic Institute, [http://geoinfo.usc.edu/gees/velacs/](http://geoinfo.usc.edu/gees/velacs/). The magnitude of the motion is close to $0.2 \, g$, while main shaking lasts for about 12 seconds (from 1 s to 13 s). Although the input earthquake motions lasts until approx. 13 seconds, simulations are continued until 120 seconds so that both liquefaction (dynamic) and pore water dissipation (slow transient) can be appropriately simulated during and after earthquake shaking (Jeremić et al., 2008).

![Input earthquake ground motions.](image)

### 505.4.4 Free Field, Lateral and Longitudinal Models

Six models were developed during the course of this study. First three models (model numbers I, II and III) were for level ground, while last three models (model numbers IV, V, and VI) were for sloping ground. First in each series of models (model I for level ground and model IV for sloping ground) were left without the second loading stage, without a pile–column system. Other four models (numbers II, III, V and VI) were analyzed for all three loading stages. Second in each series of models (models number II and V) had all displacements and rotations of pile–column top (where additional mass representing superstructure was placed) left free, without restraints. Thus, these two models represent lateral behavior...
of a bridge. Third in each series of models (model numbers III and VI) had rotations in $y$ directions fixed at the pile–column top, thus representing longitudinal behavior of a bridge. Modeling longitudinal behavior of a bridge by restraining rotations perpendicular to the bridge superstructure is appropriate if the stiffness of a bridge superstructure is large enough, which in this case it was, as it was assumed to be a post–tensioned concrete box girder, so that realistically, the top of a column does not rotate (much) during application of loads. Table 505.5 summarizes models described above.

<table>
<thead>
<tr>
<th>Case</th>
<th>Model sketch</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td>horizontal ground, no pile</td>
</tr>
<tr>
<td>II</td>
<td></td>
<td>horizontal ground, single pile, free column head</td>
</tr>
<tr>
<td>III</td>
<td></td>
<td>horizontal ground, single pile, no rotation at column head</td>
</tr>
<tr>
<td>IV</td>
<td></td>
<td>sloping ground, no pile</td>
</tr>
<tr>
<td>V</td>
<td></td>
<td>sloping ground, single pile, free column head</td>
</tr>
<tr>
<td>VI</td>
<td></td>
<td>sloping ground, single pile, no rotation at column head</td>
</tr>
</tbody>
</table>

505.5 Simulation Results

505.5.1 Pore Fluid Migration

Figures (505.9) through (505.11) show the Ru time history for up to 30 seconds, for elements (at one of Gauss point) e1, e3, e5 and e7 (refer to right side of Figure (505.6)). It is important to note that $R_u$ is defined as the ratio of the difference of initial mean and current mean effective stresses over the initial mean effective stress:

$$R_u = \frac{p'_{\text{initial}} - p'_{\text{current}}}{p'_{\text{initial}}}$$

where mean effective stress is defined as $p' = \sigma'_{kk}/3$. This is different from traditional definition for $R_u$, that uses ratio of excess pore pressure over the initial mean effective stress ($p'_{\text{initial}}$). However,
these two definitions are essentially equivalent, as soil is in the state of liquefaction for $R_u = 1$ (so that $p'_\text{current} = 0$), while there is no excess pore pressure for $R_u = 0$ (so that $p'_\text{initial} = p'_\text{current}$). However, the former definition is advocated here as it avoids the interpolation of pore pressure or extrapolation of the stresses (as the latter definition requires), since for the $u$-$p$-$U$ element, stresses are available at Gauss points while pore pressures are available element nodes. In particular, Figure 505.9 shows $R_u$ time histories for four points for models I (level ground without pile) and model IV (sloping ground without pile). It is noted that differences are fairly small. It is interesting to observe that lower layers do not liquefy as supply of pore fluid for initial void ratio of $e_0 = 0.8$ is too small, and the pore fluid dissipation upward seems to be too rapid. On the other hand, the upper soil layers do reach close to or liquefaction state ($R_u = 1$). This is primarily due to the propagation of pore fluid pressure/volume from lower layers upward (pumping effect) and, in addition to that, to a local excess pore fluid production. These results can also be contrasted with those of Jeremić et al. (2008), where similar pumping scenario has been observed. The main difference between soil used by Jeremić et al. (2008) and here is in the coefficient of permeability. Namely, here $k = 1.0 \times 10^{-4} \text{ m/s}$ was used (Čubrinovski et al., 2008; Uzuoka et al.,

![Figure 505.9: $R_u$ times histories for elements e1 (top element), e3, e5, and e7 (bottom element) Gauss point for Cases I (level ground, no pile) and IV (sloping ground, no pile).](image-url)
2008) while Jeremić et al. (2008) used $k = 5.0 \times 10^{-4} \text{m/s}$. It is important to note that other values of permeability for Toyoura sand have also been reported (Sakemi et al., 1995), but current value was chosen based on Čubrinovski (2007 –).

In addition to that, similar to Jeremić et al. (2008), sloping ground case shows larger $R_u$ spikes, since there is static shear force (stress) that is always present from gravity load on a slope. This static gravity on a slope will result in an asymmetric horizontal shear stresses in the down–slope direction during cycles of shaking. This asymmetric shear stress induces a more dilative response for down–slope shaking which will help soil regain its stiffness in the dilative parts of the loading cycles. This observation can be used to explain smaller $R_u$ spikes for the sloping ground case. Of course, this asymmetry in loading will result in larger accumulation of down–slope deformation.

While $R_u$ ratios for level and sloping ground cases are fairly similar along the depth of the model, the response changes when the pile is present. Figure (505.10) shows $R_u$ responses at four different points (along the depth) approximately midway between the pile and the model boundary, in the plane of shaking (see location of those elements in Figure (505.6) on page 2494). In comparison to behavior without the pile (Figure (505.9)), it is immediately obvious that addition of a pile with a mass on top reduces $R_u$ during shaking for the top element (e1). This is to be expected as presence of a pile–column–mass (PCM) system changes the dynamics of the top layers of soil significantly enough to reduce total amount of shear. This is particularly true for the top layers of soil as effects of column–mass tend to create compressive and extensive movements (compression when the PCM system moves toward soil and extension, and possibly even tension, when PCM system moves away from soil). However, this extension, or possible tension, is not directly observable in presented plots since array of elements where we follow $R_u$ (e1, e3, e5, e7) is some distance away from the pile–soil interface. Middle layers (e3 and e5), on the other hand, display similar response to that of Cases I and IV, as shown in Figure (505.9). It is noted that in a case with of sloping ground with pile, the $R_u$ measurements are always larger that those for level ground (this is also observed for Cases III and VI, as shown in Figure (505.11)). This is expected as presence of a pile in loose sand, and particularly the dynamic movement of a pile during seismic shaking, create an additional shearing deformation field (in the soil adjacent to the pile) that provides for additional (faster) compression of soil skeleton and thus creates additional volume of pore fluid, that is then distributed to adjacent soil (adjacent to the pile).

Particularly interesting are $R_u$ results for soil element e7, which is located below pile tip level (see Figure (505.6)). Observed $R_u$ for Case V in element e7 is significantly larger than for the same element for Case II. Similarly, simulated $R_u$ is larger than what was observed in cases without a pile (see bottom of Figure (505.9)). This increase in $R_u$ for Case V (sloping ground with pile) is explained by noting that the pile “reinforces” upper soil layers and thus prevents excess shear deformation in the upper 12.0 m
of soil (above pile tip). The reduction of deformation in upper layers of soil (top 12.0 meters) results in transfer of excessive soil deformation to soil layers below pile tip (where element e7 is located). This, in turn, results in a much larger and faster shearing of those lower loose soil layers. This significantly larger shearing results in a much higher $R_u$. Deformed shape, shown in Figure (505.12) for Case V, reinforces this explanation, showing much large shearing deformation in lower soil layers, below pile tip. Same observation can be made for Case VI, shown in Figure (505.12).

Observation similar to the above, for Cases II and V can be made for Cases III and VI, results for which are shown in Figure (505.11). One noticeable difference in $R_u$ results between cases with free column head (Cases II and V) and cases with fixed rotation column head (Cases III and VI) is in significantly larger (and faster) development of $R_u$ close to soil surface for a stiffer, no rotation column cases (Cases III and VI). This much larger $R_u$ observed in a “stiffer” PCM system setup, is due to larger shearing deformation that develops in soils adjacent to the pile during shaking. The stiffer PCM system can displace less (because of additional no rotation condition on column top) while the soil beneath is
505.5.2 Soil Skeleton Deformation

A number of deformation modes is observed for both level and sloping ground cases, with or without PCM system. Figure (505.12) shows deformation patterns and excess pore pressures in symmetry plane for all six cases over a period of eighty seconds. A number of observation can be made on both deformation patterns, excess pore fluid patterns and their close coupling.

Level Ground without Pile (Case I). Excess pore pressures and deformations in symmetry plane for level ground without a pile are shown in Figure (505.12) (I). At the very beginning (at $t = 2$ s) there is initial development of excess pore fluid pressure in the middle soil layers. This expected, as
Figure 505.12: Time sequence of deformed shapes and excess pore pressure in symmetry plane of a soil system. Deformation is exaggerated 15 times; Color scale for excess pore pressures (above) is in kN/m². Graph of ground motions used (also shown in Figure (505.8)) is placed below appropriate time snapshots and is matching for \( t = 2, 5, 10, 15, 20 \) seconds while at \( t = 80 \) seconds there is no seismic shaking.
the self weight loading stage has densified lower soil layers enough so that their response is not initially contractive enough to produce excess pore pressure. Top soil layers, on the other hand, have a drainage boundary (top surface) too close to develop any significant excess pore pressures. As seismic shaking progresses (for \( t = 5, 10 \) s), the excess pore pressure increases, and starts developing in lower soil layers as well. It should be noted that a small non-uniformity in results is present. For example, zones of variable, nonuniform excess pore pressures on the lower mid and right side for Case I at \( t = 10 \) s develop. Nonuniform mesh (many small, long elements in the middle, large elements outside this middle zone) may introduce small numerical errors in results which can be observed by slightly nonuniform results at \( t = 10 \) s and \( t = 15 \) s. It should be noted that results for excess pore pressure shown for first 13 seconds (during shaking) in Figure (505.12) (I) are transient in nature, that is, seismic waves are traveling throughout the domain (model) during shaking (first 13 seconds) and slight oscillations in vertical stresses are possible. This oscillations will contribute to the (small) non-uniformity of excess pore pressure results. After the shaking (after 15 seconds) resulting excess pore pressure field is quite uniform.

**Level Ground with Pile (Cases II and III).** Excess pore pressures and deformations in symmetry plane for models with PCM system and with two different boundary conditions at top of column (see model description in section 505.4.4) in level ground are shown in Figures (505.12) (II and III). One of the interesting observations is significant shearing and excess pore pressure generation adjacent to the pile tip. The reason for this is that pile is too short, that is, pile tip has significant horizontal displacements during shaking. Those pile tip displacements shear the soil, resulting in excess pore pressure generation. As soon as there is time for dissipation, this localized excess pore pressure dissipates to adjacent soil, and then, after shaking has ceased (at \( t = 13 \) s and later), it slowly dissipates upward. Addition of pile into the model (construction), with a highly impermeable elements (that mimic permeability of concrete) is apparent as there is a low excess pore pressure region in the middle of model, where pile is located.

**Sloping Ground without Pile (Case IV).** Excess pore pressures and deformation in symmetry plane for sloping ground without pile is shown in Figures (505.12) (IV). It is noted that initially the excess pore pressure starts developing in middle soil layers, similar to the Case I above. Bottom layers start developing excess pore pressure only after significant shear deformation occurs (at \( t = 10 \) s) at approximately 2/3 of the model depth. Lower layers have densified enough during self weight stage of loading that initial shaking is not strong enough to create excess pore water pressure, rather, those layers are fed by the excess pore pressure from above. Lower soil layers also do not develop much deformation, while middle and upper layers together develop excessive horizontal deformation.
Sloping Ground with Pile (Cases V and VI). Excess pore pressures and deformation in symmetry plane for sloping ground with PCM system are shown in Figures (505.12) (V and VI). Similar to the above cases (II and III), pile is too short and there is again excessive shearing of soil at the pile tip, suggesting large movement of that pile tip. In addition to that, pile introduces significant stiffness to upper 12 meters of soil (along the length of pile) and helps reduce deformation of those upper soil layers. Down-slope gravity load is thus transferred to lower soil layers (below pile tip) which exhibit most of the deformation. It should be noted that soil in middle and upper layers (adjacent to pile) does deform, just not as much as the soil below pile tip. The predominant mode of deformation of middle soil layers is shearing in horizontal plane, around the pile. Deformation in horizontal plane is not significant as the pile is short in this examples (as mentioned above) and does not have enough horizontal support at the bottom. The deformation pattern of a soil–pile system is such that pile experiences significant rotation, and deforms with the soil that moves down-slope. If the pile was longer, and if it had significant horizontal support at the bottom, the middle and upper soil layers would have showed more significant flow around the pile in horizontal planes.

Upper layers undergo significant settlement, as seen in Figure (505.13). This settlement is mainly caused by the above mentioned rotation of pile–soil system, where soil in general settles (compacts) but also undergoes differential settlement, between left (up-slope from pile) and right (down-slope from pile) side of the model. As significant shearing with excess pore pressure generation develops in lower soil layers, below pile tip, those lower layers contribute to most of down-slope horizontal deformation. In a sense, all the demand from down-slope gravity forces and seismic shaking is now responded to by lower soil layers, which contribute to most of the excess pore pressure generation and consequently, to most of the soil deformation. Soil surface horizontal deformation is thus strongly influenced by significant shearing of the bottom layers and by rotation of the middle and upper soil layers with the pile. It is interesting to note that the largest settlement is observed just down-slope from pile for Cases V and VI.

505.5.3 Pile Response

Figure (505.14) shows bending moment envelops for pile–column–mass (PCM) system for all four cases (II, III, V and VI). It should be noted that bending moment diagrams are plotted on compression side of the beam–column. A number of observations can be made about bending moment envelopes. For cases with free pile head (shaking transverse to the bridge main axes, Cases II and V) the maximum moments are attained in soil, at depths of approximately \(0.6D - 1.2D\), where \(D = 1.0\) m in this case) is the pile diameter. Opposed to that are cases for PCM systems with restricted rotations at the pile top which (Cases III and VI), which, of course feature largest moment at the column top. Maximum bending moments for section of PCM system in soil (pile) in these two cases are now attained at the...
depth of approximately $1.8D - 2.0D$.

It is noted that bending moment envelopes are mostly symmetric. Slight non-symmetry is introduced for cases on sloping ground (Case V and VI). It is also noted that moments do exist (are not zero) all the way to the bottom of the pile. Theoretically, moments should be zero at the pile tip, but since physical volume of the pile is considered (see note on that in section 505.4.2 and Figure (505.7)), differential pressure on pile bottom from soil will produce small (non-zero) moments even at the pile tip. More importantly, non-zero moments at the bottom and along the lower part of the pile show that pile is indeed too short, and thus changing curvatures are present along the whole length of the pile.

### 505.5.4 Pile Pinning Effects

Piles in sloping liquefying ground can also be used to resist movement of soil (all liquefied or liquefied with hard crust on top) down-slope. For models developed in this paper, pile pinning effect can be investigated for Cases IV, V and VI. In particular, deformation of sloping ground without the pile (Case IV) can be compared with either of the cases of piles in sloping ground, Cases V and VI. It is very important to note, again, that models developed here had relatively short pile, and that major soil shearing developed below the pile tip. This apparent shortcoming of a short pile results in reduced pile pinning capacity, thus reducing the down-slope movement by only approximately half, from 0.35 m (Case IV) to 0.22 m (Case V) and to 0.18 m (Case VI) as seen in Figure (505.15). It would have been expected that, had the pile been longer and had it penetrated in deeper, non-liquefiable layers, it would
Figure 505.14: Envelope of bending moments for pile–column system for Cases II, III, V and VI.

have reduced down–slope movement of the soil to a much larger extent. However, had the pile been longer and had it penetrated non-liquefiable layers, it would have had a much firmer horizontal support at the bottom and would have thus attracted much larger forces too, potentially leading to pile damage and yielding.
Figure 505.15: Down-slope movement at the ground surface (model center) for Cases IV (no pile), V and VI (with pile–mass system).
Chapter 506

Slope Stability in 2D and 3D

(In collaboration with Dr. Nima Tafazzoli)
506.1 Chapter Summary and Highlights

506.2 Introduction

Presented here is a brief overview of a work on determining factors of safety for a 3D slope for one of the prominent dams in the midwest, for different saturation and water level conditions.

506.3 Dam Section Geometry

Figures 506.1 and 506.2 shows the satellite picture of location

Figure 506.1: Dam satellite picture. Upstream (south facing) three dimensional slope at connection of the embankment and concrete gravity dam is evaluated for potential stability problem.

Figure 506.3 shows the sonar data for water depth measurements at the location of a 3D slope in question.

An interesting topographical feature resembling a valley should be noted at the toe of (beginning of) wrap around section of upstream section of the embankment. That topographical feature is also apparent in a photo taken during Wold Creek Dam construction in 1948, show in Figure 506.4. Figure 506.5 shows rock surface under the embankment and alluvial fill, that serves as base rock foundation for both the alluvium and the embankment above. Figures 506.6, 506.7 and 506.8 show sections of the curved, 3D slope of the upstream embankment next to the concrete dam section. In particular, Figure 506.6 shows a dam section perpendicular to the dam axes. Note a significant extent of the alluvium that was left in place during dam construction. Similarly, the alluvium is present in both section inclined at 45° to dam axes (Figure 506.7) and a section parallel to the dam axes (Figure 506.8).

Analyzed 3D, curved section of the upstream embankment is also shown in a photograph in Figure 506.9, taken on morning of April 21st, 2010. Note a significant length/extent of the two sheet pile walls (one running along the length of work platform and the other one at the end of work platform, next to...
Figure 506.2: Dam satellite picture with location of 3D slope under consideration for potential stability problem.

Figure 506.3: Dam sonar data for water depth for the analyzed 3D slope.

concrete dam section).
Figure 506.4: Dam during construction, 1948. View toward the upstream curved slope. Note shallow valley where the straight section of pole line ends, approximately at the end or straight slope, toward the beginning of curves slope section (Photo courtesy of Mike Zoccola, USACE).
Figure 506.5: Dam rock surface data under the embankment and alluvial fill (under analyzed 3D slope), used for defining FEM model boundary conditions.

Figure 506.6: Dam section perpendicular to the dam axes for analyzed 3D slope.
Figure 506.7: Dam section inclined at $45^\circ$ to dam axes for analyzed 3D slope.

Figure 506.8: Dam section parallel to the dam axes for analyzed 3D slope.
Figure 506.9: Dam view toward the curved, "wrap" section, upstream side.
506.4 Finite Element Modeling

Analysis of 2D and 3D slope stability problem was performed using finite element method. In particular, the strength reduction method was used in conjunction with 2D and 3D finite element models to assess factors of safety. Main focus was on determining the difference between 2D and 3D factors of safety.

506.4.1 Material Models

During initial teleconference, in early April, it was decided to perform two dimensional and three dimensional analysis using finite element method for two limiting cases of material behavior:

- fully drained material behavior, with material defined by a friction angle and no cohesion,
- fully undrained material behavior, with material defined by an undrained shear strength.

Parameters for drained material modeling are given in table 506.1 below. In addition to that table 506.2 below, defines parameters for undrained material modeling. Those material parameters were agreed upon after few and a discussion. It should be noted that USACE has initiated a testing program that will determine material parameters of the embankment and the alluvium with higher certainty.

Two material models were used for analysis. For drained analysis, a Drucker Prager material model was used. This model is described in some detail in Section 3.5.7 of my lecture notes Jeremić et al. (1989-2021) (available online through my web site). It is important to note that single value of friction angle was used, thus rendering friction angle for both compression and extension the same. While there might be an influence of the difference of friction angle for compression and extension, such difference was not analyzed here. For undrained analysis a total stress, von Mises model was used. This model is also described in some details in Section 3.5.6 of my lecture notes Jeremić et al. (1989-2021). Both models used here feature perfectly plastic behavior after yielding, with associated plastic flow.

<table>
<thead>
<tr>
<th>Material</th>
<th>Unit Weight [pcf]</th>
<th>$\phi$ [deg]</th>
<th>$S_u$ [psf]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embankment Fill</td>
<td>126.7</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Alluvial Soil</td>
<td>116</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>Random Fill</td>
<td>120</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Riprap</td>
<td>110</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>Shot Rock Fill</td>
<td>145</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>Platform Extension Fill</td>
<td>120</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

506.1: Drained strength material parameters.
Table 506.2: Revised undrained strength material parameters.

<table>
<thead>
<tr>
<th>Material</th>
<th>Unit Weight [pcf]</th>
<th>$\phi$ [deg]</th>
<th>$S_u$ [psf]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embankment Fill</td>
<td>126.7</td>
<td>0</td>
<td>2000</td>
</tr>
<tr>
<td>Alluvial Soil</td>
<td>116</td>
<td>0</td>
<td>1000-1500</td>
</tr>
<tr>
<td>Random Fill</td>
<td>120</td>
<td>0</td>
<td>500</td>
</tr>
<tr>
<td>Riprap</td>
<td>110</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>Shot Rock Fill</td>
<td>145</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>Platform Extension Fill</td>
<td>120</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

506.4.2 Two Dimensional Models

Three two dimensional (2D) models were developed in order to test the effects mesh size has on quality of simulations. Developed 2D mesh was based on perpendicular cross section, shown in Figure 506.6. Mesh for a 2D model, shown in Figure 506.10, represents a very coarse mesh, which, if higher order elements are used (with displacement interpolation higher than linear) might actually work quite well. However, with linear interpolating displacements, this mesh is clearly of low quality. However, this mesh was used as a first iteration, and to gain initial insight into 2D behavior.

![Figure 506.10: Coarse mesh model used for the 2D upstream section of the Dam.](image)

Meshes shown in Figures 506.11 and 506.12, represent a refinement of the first mesh. Models using both meshes were tested for both drained and undrained material models. Results for both fine mesh (Fig. 506.11) and finest mesh (Fig. 506.12) were the same for all practical purposes (differences were negligible for the purpose of analysis), hence it was decided to use fine mesh (Fig. 506.11) for all
Figure 506.11: Fine mesh model used for the 2D upstream section of the Dam. This mesh was chosen as appropriate enough for all subsequent 2D analysis.

Figure 506.12: Much finer mesh model used for the 2D upstream section of Dam. Results obtained this mesh were the same as previous mesh (shown in Figure 506.11), leading to the choice of the previous mesh (Figure 506.11) for all 2D analysis.

It is important to note that all 2D meshes were actually developed using 3D, brick elements (in this case a linear interpolation, 8 node bricks). Appropriate boundary conditions were used to prevent out of plane displacements. While this approach uses extra resources (use of 3D element for a 2D problem) it allowed us to use the very same element and material models for both 2D and 3D problems, thus removing modeling uncertainty and emphasizing accuracy of determination of differences between 2D
and 3D slope stability. The finite element used for analysis in 2D (and later in 3D) was based on an eight node brick element, described in some more details in Section 2.3 of my lecture notes Jeremić et al. (1989-2021).

Mesh in Figure 506.11 has 250 brick finite elements while finest mesh in Figure 506.12 has 1000 brick finite elements. Boundary conditions for the 2D mesh were such that right vertical boundary (at the current Barrier wall) was allowed to move down but not horizontally. Lower mesh boundary (contact with base rock) was fully supported, while left vertical boundary (far into alluvium) was also allowed to move vertically but not horizontally. Work platform was modeled using loads, instead of extending the mesh to include the shape of the platform. This allowed for additional modeling flexibility, without the loss of accuracy. Loading tests with and without the work platform show that its influence on calculated factors of safety is negligible. Figure 506.13 shows a 2D model with location of work platform, water level at 680 ft, boundary conditions and the extent of alluvium layer.

Figure 506.13: A 2D model with boundary conditions, water level at 680 ft and the work platform as load.

506.4.3 Three Dimensional Models

The Dam is a full three dimensional solid, consisting of foundation rock, concrete section and soil embankment section with barrier wall(s) (old and new one when finished). Ideally we would model the complete foundation-concrete-embankment-wall system with porous solid (soil skeleton), pore fluid
Particular focus on assessing 2D versus 3D slope stability for the upstream, curved (wrap around) section, as well as the very short time frame for this project, dictated development of small part of the complete model. The most important feature of the 3D model was to use optimal model size (extent) so that all important features are properly captured. This was done by developing 3D model for upstream curved, wrap slope in two stages. First stage comprised development of the conical portion of mesh, which was then extended, using a straight second stage, extending the model into the embankment by 200 ft (60 meters).

**Conical Section of a 3D Slope** Three sections shown in Figures 506.6, 506.7 and 506.7 were used to develop three dimensional mesh. Conical wrap around part of the mesh features 1344 brick elements and is shown in Figure 506.14.

![Conical section of the 3D upstream slope of the Dam, 3D mesh.](image)

Boundary conditions are such that two vertical sections (one adjacent to the concrete monolith and the one adjacent to straight embankment sections have free vertical and in plane horizontal displacements while no out of plane displacement. A small quarter cylinder boundary was also developed at the top of the curved, wrap around section. This small part of boundary (radius of 2 meters) was necessary to properly mesh radial directions of the 3D model. Boundary conditions at that location are such that they prevent radial deformation while allowing vertical and tangential movement. This is achieved using short, stiff truss elements.
At the lower end of the model (well into alluvium), curved, radial section of the model is supported in such a way that radial deformation is prevented while vertical and horizontal tangential movement is allowed, again achieved using short, stiff truss elements.

![Figure 506.15: Curved section of the 3D upstream slope of the Dam, 3D mesh, top view. Note the difference in extent of the actual embankment (slightly different color of elements, alluvium is represented by last two layers of elements on the outskirts of model. Also note a small cylindrical section close to center of conical slope, that was used to control meshing.](image)

**Conical and Straight Section of a 3D Slope**  While conical section of the slope represent accurately the curved section of the dam, influence of the straight section cannot be neglected. A section (200 ft, 60 meters) of a straight embankment was added to a curved section in order to have a more realistic modeling of the complete upstream 3D embankment. Figures 506.16 and 506.17 show extended 3D mesh for curved and part of straight section of the upstream slope.

Boundary conditions were similar to the previous case for conical part of mesh, except where mesh extension was applied. Extended mesh was fully supported at the bottom, while vertical faces were supported in such a way not to have out of plane displacements while in plane (horizontal and vertical) displacements were left free. Loads from work platform and from water in the lake were applied as appropriate nodal forces.
Figure 506.16: Curved and straight section of the 3D upstream slope of the Dam, 3D mesh.

Figure 506.17: Curved and straight section of the 3D upstream slope of the Dam, 3D mesh.
506.4.4 Modeling Issues

Mesh Size  Multiple mesh sizes were tested to alleviate any mesh dependency of the computed solution close to stability loss (close to but not into the development of localized deformation).

Following the Equilibrium Path  Numerical integration of elastic-plastic constitutive equations was performed using both explicit and implicit algorithms (see Jeremić et al. (1989-2021), Sections 3.3 and 3.4). This was necessary in order to verify the solution accuracy. For explicit constitutive computations, explicit global solution scheme was used (see more detailed description of equilibrium following path algorithms in my lecture notes Jeremić et al. (1989-2021), Chapter 7). For implicit constitutive integrations, Newton method (cf. Dennis and Schnabel (1983), and Jeremić et al. (1989-2021), Chapter 7) was used on both constitutive and global solution levels. Global solution advancement was constrained using both load control (appropriate for factor of safety stability simulations) as well as displacement control (in order to validate load control solutions). Detailed discussion on both load and displacement control algorithms is given in lecture notes Jeremić et al. (1989-2021), Sections 7.3 and 7.4.

Strength Reduction Method  Strength reduction method is often used with finite elements to assess factors of safety for slopes, foundations and other problems in geotechnical engineering where elastic-perfectly plastic failure is expected Duncan (1996) Griffiths and Lane (1999). Strength reduction method can be performed in two ways:

- Apply self weight to a model where strength parameters are reduced by Factor of Safety (FS). Perform analysis number of times for larger and larger FS, until the finite element system is not in equilibrium, that is external forces cannot be balanced by internal forces (see Section 7.1 in Jeremić et al. (1989-2021)). This approach has the benefit of not being mesh dependent, that is, for elastic – perfectly plastic material models (without softening), dependence of solution on mesh refinement does not become an issue (Lu et al., 2009). One potential drawback is that since there is no softening involved, deformation will not localize into localized zone, rather failure mode is somewhat diffuse, however still following proper displacement of blocks of material as system becomes unbalanced.

- Apply self weight to a model with full strength material parameters, and then gradually start softening the material parameters, by dividing them with FS. Perform reduction in parameters, until there is loss of equilibrium, that is until external forces cannot be balanced by internal forces (again see Section 7.1 in Jeremić et al. (1989-2021)). This approach might have a potential problem in that the mesh size/refinement will affect the solution, as material models involved are
softening, and different meshes will create different localization patterns and hence lead to different solutions. The benefit of this approach is that deformation will indeed localize in a very thin band, thus resembling the limit equilibrium approach that is popular for slope stability problems, however, such deformation patterns are mesh size/refinement dependent and thus not unique.

In this work, the first approach is used, that is self weight computations are performed for a number of models where strength parameters are reduced by FS, until such FS for which equilibrium between external and internal forces cannot be achieved.

**Variable Water Levels**  Comparison of 2D vs 3D stability for two water elevation was originally suggested. Discussions during a meeting at a Dam site, further clarified conceptual problems with two water elevation, namely lake at 680 ft and water table in the embankment at 720 ft. For any water table that is above the lake level, there will exist a water flow (seepage) within the embankment toward the (lower level) lake water. Such seepage will create a seepage force \( f_s = \gamma_w \frac{(dh)}{(dl)} \) where, \( \gamma_w \) is the unit weight of water, and \( i = \frac{(dh)}{(dl)} \) is the hydraulic gradient. Clearly, with seepage from embankment toward the lake, such force destabilizes the embankment (slope). Finite Element formulation and tools that can model and simulate such fully coupled system (transient analysis of deforming porous soil with moving pore fluid) are available (Cheng et al., 2007; Jeremić et al., 2008; Cheng and Jeremić, 2009a,b), however material parameters for soil permeability as well as the extent of phreatic surface need to be determined. Due to unavailable test data for permeability of embankment soil and due to (high) uncertainty in the extent of phreatic surface (number of piezometric measurements placing the water table in the embankment at 720 ft were highly questionable, as discussed by the panel), only single phase soil (either effective stress or total stress) analysis were performed. This decision has no effect on analysis for cases where both lake and phreatic line in the embankment are at 680 ft. However, such decisions affects cases with different water levels (lake at 680 ft, phreatic line at):

- Fully undrained case with lake level at 680 ft, embankment phreatic level at 720 ft, was analyzed and in fact provides (based on theory of undrained soil behavior) the same factors of safety as no seepage case, that is, same water level in lake and embankment. This is due to the fact that for total stress analysis, pore fluid pressure does not influence the shear strength. In addition to that, with the assumption of fully undrained conditions, coefficient of permeability is nonexistent, there is no seepage, and hence there is no seepage force. In reality coefficient of permeability for soil is never really zero, however this case was treated as an extreme case, used purely for checking differences in factors of safety between 2D and 3D slope stability.

- Fully drained case with lake level at 680 ft, embankment phreatic level at 720 ft, was not analyzed
as neglecting seepage force would place calculated factors of safety on the unsafe side. While such analysis can be run using either buoyant weight approach (however somewhat inconsistent as noted by Duncan (1996)) or by mixing total stress and effective stress approach (more inconsistent) it was decided not to simulate this case.

### 506.5 Results: Factors of Safety

A large number of computations on a number of models were performed.

**Factors of Safety for 2D Models**

- Drained, Lake/Embankment at 680 ft,
  \[ FS_{2D} = 1.89 \]

- Undrained, Lake/Embankment at 680 ft, alluvium weak of most likely \( S_u = 1000 \text{ psf} \),
  \[ FS_{2D} = 2.22 \]

- Undrained, Lake/Embankment at 680 ft, alluvium at most likely value \( S_u = 1500 \text{ psf} \),
  \[ FS_{2D} = 2.50 \]

**Factors of Safety for Extended 3D Models**

- Drained, Lake/Embankment at 680 ft,
  \[ FS_{3D} = 1.78 \]

- Undrained, Lake/Embankment at 680 ft, (alluvium weak of most likely \( S_u = 1000 \text{ psf} \)),
  \[ FS = 2.0 \]

**Comparison of 2D vs 3D Factors of Safety**

- Drained, lake/water in embankment at 680 ft,
  \[ FS_{2D} = 1.89 \text{ vs } FS_{3D} = 1.78 \]
  **FS reduced by 5.8 %**.

- Undrained, Lake at 680 ft, embankment at 680 ft
  (alluvium weak of most likely \( S_u = 1000 \text{ psf} \)),
  \[ FS_{2D} = 2.22 \text{ vs } FS_{3D} = 2.00 \]
  **FS reduced by 9.91 %**.
• Undrained, Lake at 680 ft, embankment at 720 ft
  (alluvium weak of most likely $S_u = 1000$ psf),
  $FS_{2D} = 2.22$ vs $FS_{3D} = 2.00$
  FS reduced by 9.91%.

506.6 Uncertainty of Results

Full probabilistic analysis for large scale, elastic-plastic finite element models is currently within reach (Sett et al., 2011a) and will probably become a standard simulation tool within next 5 to 10 years. However, for such an analysis, a(n) (extensive) site characterization program is necessary, if probabilistic simulations are to be useful (Sett and Jeremić, 2009). Alternatively, one can use a simplified methodology, described in detail by Duncan (2000a), that can give sensible estimates of mean and standard deviation of common results obtained in soil mechanics. However, in view of lack of consistent material properties for the embankment further work on estimating uncertainty of factors of safety was discontinued. It is important to note that a program was initiated by the USACE to perform extensive testing of embankment soil and upon completion of that program, it will be possible to perform a simplified (and even a more accurate, full probabilistic analysis, as mentioned above) estimation of influence of material uncertainty on obtained factors of safety.

506.7 Conclusion

The main purpose of this study was to investigate changes in factors of safety for failure between 2D and 3D slope problems for curved part of the upstream section of a Dam embankment. Detailed models (both 2D and 3D) were developed for the upstream section of Wold Creek Dam and were analyzed. Strength reduction method was used to assess factors of safety in both 2D and 3D. It was shown that for a number of different cases (drained or undrained soil, lake at 680 ft and water table in the embankment either at 680 ft or at 720 ft for undrained case) the factor of safety is reduced in 3D when compared to 2D. Such reduction, however, was not significant (up to approx. 10%).

Of particular importance for this factor of safety comparison (2D vs 3D) was the robustness of simulations and a number of methods were used to ensure that obtained factors of safety, determined as failure to converge upon strength reduction, were due to loss of equilibrium, and not due to numerical problems leading to loss of convergence.

Appendix given below, contains two additional sections, describing numerical tool used in factor of safety computations (available in public domain) and deformation patterns close to failure for select
506.8 Displacement Patterns

Hypothetical displacement patterns are presented below in Figures 506.18, 506.19 and 506.20. It should be noted that displacements that results from simulations of stability by strength reduction method are not necessarily the failure modes, rather, they are just a side-product of simulation. However, it is instructive to inspect those displacements just before equilibrium is lost in order to gain a better understanding of potential failure patterns.

Figure 506.18 shows a 2D displacement pattern for a drained case with lake and embankment water level at 680 ft where $FS_{2D} = 1.89$. Noted is a clear rotating pattern of the slope just before loss of equilibrium.

Figure 506.18: Drained case, lake and embankment water lever 680 ft, $FS_{2D} = 1.89$.

Figure 506.19 shows a 2D displacement pattern for an undrained case with more likely (stronger) alluvium ($S_u = 1000$ psf). Note that while displacement pattern is somewhat similar to the previous case, the block movement on top of alluvium is much reduced, and main failure is through rotation mechanism.

Figure 506.20 shows a plan view of a 3D displacement pattern (vectors) for an undrained case with limiting weak alluvium and lake at 680 ft. Noted is the dominant 3D deformation pattern by which the curved and straight sections of the slope rotate around vertical axes and move toward the lake with direction biased toward dam axes. It is important to note that there is no single 2D section of the 3D
Figure 506.19: Undrained, L680 E680, Most Likely to Weak Alluvium, $FS_{2D} = 2.22$.

slope that can be used to model such failure as failure mode is fully three dimensional.
Figure 506.20: Plan view of a 3D undrained case, displacement vectors, lake and embankment at 680 ft, $FS_{3D} = 2.00$. 
Chapter 507

Concrete Structures


(In collaboration with Dr. Han Yang and Dr. Hexiang Wang)
507.1 Chapter Summary and Highlights

507.2 Concrete Wall/Membrane

507.2.1 Introduction

OECD organized project to investigate modeling of concrete walls that have alkali - silica reaction (ASR) has been going on for few years. Our modeling and simulation group, based at the University of California at Davis, in California, USA, and at the Lawrence Berkeley National Laboratory at Berkeley, in California, USA, has joined this effort only at the beginning of 2018. We have gotten project description and already existing reports from participants, in January/February of 2018. We managed to quickly make a model of the wall and to calibrate parameters for a concrete model that we use in our modeling and simulation program.

We rely on and used our program, called the Real-ESSI Simulator (Realistic Modeling and Simulation of Earthquakes, and Soils, and Structures and their Interaction), that is also known as the MS ESSI Simulator\(^1\) (Jeremić et al., 1988-2021) (http://real-essi.info/). Real-ESSI Simulator is a software, hardware and documentation system for high fidelity, high performance, time domain, nonlinear/inelastic, deterministic or probabilistic, 3D, finite element modeling and simulation of (a) statics and dynamics of soil, (b) statics and dynamics of rock, (c) statics and dynamics of structures, (d) statics of soil-structure systems, and (e) dynamics of earthquake-soil-structure system interaction.

The Real-ESSI Simulator systems is used for the design and assessment of static and dynamic behavior of infrastructure objects, including buildings, bridges, dams, nuclear installations, tunnels, etc. For design, multiple linear elastic load cases can be combined and design quantities, sectional forces exported for design and cross section dimensioning. For assessment, realistic inelastic load staged analysis is performed, with all the inelastic components properly modeled, as listed below, and with all the simulation, algorithmic features available, as listed below. Analysis is performed in order to assess safety and economy of objects. The work on Real-ESSI Simulator is based on a philosophy that aims to develop modeling and simulations that inform and predict rather than (force) fit.

507.2.1.1 Motivation

The alkali - silica reaction (ASR), is a reaction that occurs over time in concrete, between the alkaline cement paste and the silica found in many common aggregates, triggered by the presence of moisture. The ASR does volumetric expansion of interface between cement paste and the aggregate, thus resulting

\(^1\)The Real-ESSI Simulator was developed in collaboration and with financial support from the US-DOE, US-NRC, US-NSF, CNSC-CCSN, Caltrans, etc.
in potentially damaging tensile stress within concrete. A comprehensive study of this effect on concrete properties has been done at University of Toronto, funded by Canadian Nuclear Safety Commission (CNSC). Five shear walls, two regular (REG A, REG B) and three ASR (ASR A1, ASR B1, ASR B2) along with their control specimens were tested as a part of Phase 1, Phase 2, and Phase 3 of the investigation program [ref Prof. Sheikh presentation].

It was proven that ASR causes changes in mechanical properties of the concrete, introducing inaccuracies if conventional analysis is used, without taking into account ASR. Beyond a certain age, modulus of elasticity and tensile strength of ASR concrete decrease, in contrast to regular concrete. Observed is degradation of the ductility in ASR specimens. While concrete expands, the bonding strength between concrete and rebars decreases causing drop in ductility. However, despite these changes in concrete, tested ASR shear wall showed higher peak shear strength compared to regular concrete.

This newly understood behavior of ASR concrete necessitates development of reliable numerical simulation, to be used for the design of new structures as well as to predict behavior of existing ones.

In order to model behavior of the examined wall specimens, Real-ESSI Simulator, (Jeremić et al., 1988-2021) was used.

Plane stress finite element with an inelastic plane stress material model for concrete was used for modeling of shear wall specimens. All inelastic material models and finite elements inside the Real-ESSI Simulator feature accurate energy dissipation modeling and calculation (Yang et al., 2018, 2019).

Results from ASCET benchmark tests are used for validation of shear behavior for plane stress inelastic reinforced concrete wall tests.

507.2.2 Model Development

507.2.2.1 Model Mesh

Finite element model includes all relevant parts of the experimental setup, as shown in Figures 507.3 and 507.2.

Beam slabs and steel plates are modeled using 27NodeBrick element, while steel bolt is represented as a single truss element. The shear wall is modeled using nonlinear layered plane stress elements. For the web part of the wall, the elements have a horizontal rebar layer, a vertical rebar layer, and an unconfined concrete layer. For the flanges (columns) of the shear wall, the elements have an additional layer of confined concrete. It is emphasized that the main wall is really made from unconfined concrete, and the only actually confined concrete is within concrete flanges. Detailed rebar plan and model dimensions are shown in Fig 507.3.

The bottom of the model is restrained in all directions, while the lateral sides of the bottom beam slab
Figure 507.1: 3D rendered view and finite element mesh of the reinforced concrete shear wall model.
are restrained in direction of imposed motion. Since the shear wall consists of 2D plane stress elements the out-of-plane displacement is also precluded. The sides of the top beam slab are also restrained to have the same displacement, which is important to represent the boundary conditions of the physical experiment. Initial model included inelastic contact/interface elements (stick-slip and gap open and close) at the bottom boundary. However it was concluded that there will be no slip and there is no gap opening so these elements were removed in order to speed up computations.

To correctly simulate the loading process of the experiment, four loading stages are applied:

1. Self-weight loading is applied to the whole model.

2. To represent post-tensioned force in the truss (bolt) element, the truss is stretched so that an adequate force is obtained and after that the bottom of the bolt is fixed.

3. Two-point vertical loading is applied to the top steel plate.

4. Cyclic horizontal loading is applied (using displacement control) to the sides of the top beam slab.
507.2.2.2 Plastic Damage Concrete Material Model

Details for this model are given in Section 104.9.

507.2.2.3 Uniaxial Steel Material Model

Details for this model are given in Section 104.9.

507.2.2.4 Material Model Parameters

There are two approaches to determine the ASR affected concrete model parameters. The first choice is to obtain concrete samples directly from the existing structure. Those samples can be tested in the laboratory to determine mechanical properties of the concrete material. This is a preferred way to obtain material properties.

In this project, the material properties for ASR affected concrete were obtained using laboratory, for
concrete aged to 260, 610, and 995 days, as shown in Table 507.1. These values are used in the material models for finite element simulations.

<table>
<thead>
<tr>
<th>Wall</th>
<th>Age (Days)</th>
<th>Compressive Strength (MPa)</th>
<th>Tensile Strength (MPa)</th>
<th>Elastic Modulus (MPa)</th>
<th>Expansion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR A1</td>
<td>260</td>
<td>63.7</td>
<td>3.24</td>
<td>35750</td>
<td>0.190</td>
</tr>
<tr>
<td>ASR B1</td>
<td>610</td>
<td>67.1</td>
<td>N/A</td>
<td>32600</td>
<td>0.215</td>
</tr>
<tr>
<td>ASR B2</td>
<td>995</td>
<td>63.0</td>
<td>3.18</td>
<td>28100</td>
<td>0.223</td>
</tr>
</tbody>
</table>

The second approach is to estimate change of material properties, future material properties, based on currently measured material properties and empirical correlations. In what follows, it is demonstrated how to predict the mechanical properties of the ASR concrete at 610 and 995 days using the values at 260 days.

First, according to data provided for ASCET II workshop, the maximum ASR expansions of the concrete prisms in longitudinal and transverse directions are approximately 0.25% and 0.28%. For simplicity, an average value of 0.265% is chosen.

Estimations of ASR expansion rates at 260, 610, and 995 days are needed. According to various ASR models (Charlwood et al., 1992; Saouma and Perotti, 2006), the rate of expansion after about 365 days is very small. So it is reasonable to use the maximum value of expansion (0.265%) for estimating mechanical properties of ASR concrete at 610 and 995 days. At 260 days, the expansion is estimated to be 0.23%.

The correlation between normalized mechanical properties of ASR concrete and rate of expansion/swelling from Capra and Sellier (2003) is used. At 0.265% swelling, at 365 days, the normalized compressive strength is 0.60 while the normalized tensile strength is 0.55. The normalized elastic modulus ranges from 0.60 to 0.70.

On the other hand, at 0.23% swelling, at 260 days, the normalized compressive strength is 0.65 while the normalized tensile strength is 0.60. The normalized elastic modulus is approximately 0.70. Therefor, from 260 days to 365 and consequently at 610 and 995 days, the compressive and tensile strengths both decrease approximately 92%. Similarly, the elastic modulus decreases approximately 85%.

Using these values, for ASR concrete older than 365 days, the compressive strength is 58.6MPa, the tensile strength is 2.98MPa, and elastic modulus 30387MPa. Note that these values match well with the experimental results shown in Table 507.1. It is important to note that concrete is a composite material and that expansion of mass of concrete, within confines of reinforcing, can produce increase in confinement, which can have beneficial effects on concrete behavior.
507.2.3 Modeling of Energy Storage and Dissipation

Described in this section is modeling and calculation of energy storage and dissipation within concrete and steel. This section is based on recent work by Yang et al. (2018, 2019).

507.2.3.1 Plastic Damage Concrete Material

The Helmholtz free energy potential postulated for the plastic damage concrete material has the form:

\[ \psi(\epsilon_{ij}, \epsilon^p_{ij}, d^+, d^-) = (1 - d^+)\psi^+_0(\epsilon_{ij}, \epsilon^p_{ij}) + (1 - d^-)\psi^-_0(\epsilon_{ij}, \epsilon^p_{ij}) \]  
(507.1)

where \( \psi^+_0 \) and \( \psi^-_0 \) are tensile and compressive parts of the elastic free energy (strain energy), that are defined as:

\[ \psi^+_0(\sigma_{ij}(\epsilon_{ij}, \epsilon^p_{ij})) = \frac{1}{2}\sigma^+_{ij}D^{-1}_{ijkl}\sigma_{kl} = \frac{1}{2}\sigma^+_{ij}(\epsilon_{ij} - \epsilon^p_{ij}) \]  
(507.2)

\[ \psi^-_0(\sigma_{ij}(\epsilon_{ij}, \epsilon^p_{ij})) = \frac{1}{2}\sigma^-_{ij}D^{-1}_{ijkl}\sigma_{kl} = \frac{1}{2}\sigma^-_{ij}(\epsilon_{ij} - \epsilon^p_{ij}) \]  
(507.3)

The rate of strain energy can be calculated from:

\[ \dot{E}_S = \dot{\psi}^+_0 + \dot{\psi}^-_0 = \sigma_{ij}(\dot{\epsilon}_{ij} - \dot{\epsilon}^p_{ij}) \]  
(507.4)

Taking the time derivative of Equation 507.1 gives the rate form of Helmholtz free energy:

\[ \dot{\psi} = \frac{\partial \psi}{\partial \epsilon_{ij}}\dot{\epsilon}_{ij} + \frac{\partial \psi}{\partial \epsilon^p_{ij}}\dot{\epsilon}^p_{ij} + \frac{\partial \psi}{\partial d^+}\dot{d}^+ + \frac{\partial \psi}{\partial d^-}\dot{d}^- = (\sigma_{ij}\dot{\epsilon}_{ij} - \sigma_{ij}\dot{\epsilon}^p_{ij}) + (\dot{\psi}^+_0\dot{d}^+ + \dot{\psi}^-_0\dot{d}^-) \]  
(507.5)

where the first term is the rate of strain energy given by Equation 507.4.

For a decoupled material model, the Helmholtz free energy can be decomposed into elastic and plastic parts (Collins and Houlsby, 1997). Subtracting the elastic part of the free energy (strain energy) from Equation 507.5 gives the rate form of the plastic free energy:

\[ \dot{E}_P = \dot{\psi} - \dot{E}_S = -\dot{\psi}^+_0\dot{d}^+ - \dot{\psi}^-_0\dot{d}^- = -\frac{1}{2}(\sigma^+_{ij}\dot{d}^+ + \sigma^-_{ij}\dot{d}^-)(\epsilon_{ij} - \epsilon^p_{ij}) \]  
(507.6)

The plastic energy dissipation \( D_P \) is defined as the difference between plastic work and plastic free energy (Farren and Taylor, 1925; Taylor and Quinney, 1934). The rate of plastic dissipation can be expressed as:

\[ \dot{D}_P = \sigma_{ij}\dot{\epsilon}^p_{ij} - \dot{E}_P = \sigma_{ij}\dot{\epsilon}^p_{ij} + \frac{1}{2}(\sigma^+_{ij}\dot{d}^+ + \sigma^-_{ij}\dot{d}^-)(\epsilon_{ij} - \epsilon^p_{ij}) \]  
(507.7)

It can be proven that the rate of plastic dissipation given by Equation 507.7 is always non-negative. This means that the concrete material model used in this study satisfies the Clausius-Duhem inequality, which represents a form of the second law of thermodynamics.
Combining Equation 507.4, 507.6, and 507.7, the rate form of energy balance in this concrete material model is achieved:

\[
\dot{E}_S + \dot{E}_P + \dot{D}_P = \sigma_{ij} \dot{\epsilon}_{ij} \tag{507.8}
\]

### 507.2.4 Uniaxial Steel Material

The energy computation procedure for the uniaxial steel model is shown in Figure 507.4. Note that the only difference between the monotonic loading branch (Figure 507.4(a)) and the cyclic loading branch (Figure 507.4(b)) is that the strain reversal point \( c \) is at the origin \( o \) in the monotonic case. So the following explanation of the proposed energy computation method applies to both monotonic and cyclic loading scenarios.

![Figure 507.4: Energy computation of uniaxial steel fiber: (a) Monotonic loading branch; (b) Cyclic loading branch.](image)

Firstly, the elastic strain energy density \( E_S \) is defined in accordance with the classic assumption that it is only a function of current stress state of the material, which yields:

\[
E_S = E_S(\sigma) = \frac{1}{2E_0} \sigma^2 \tag{507.9}
\]

where \( E_0 \) is the initial stiffness of the material.

Graphically, the elastic strain energy density of the material shown in Figure 507.4 at states \( a \) and \( b \) are the triangular areas \( afd \) and \( bge \). Then the incremental form of Equation 507.9 is simply:

\[
dE_S = \frac{1}{E_0} \sigma \, d\sigma \tag{507.10}
\]
Next, the incremental plastic dissipation density $D_P$ from state $a$ to $b$ is assumed to be the triangular area $abc$:

$$dD_P = \frac{1}{2}[(\sigma - \sigma_r)de - (\epsilon - \epsilon_r)d\sigma]$$  (507.11)

This assumption ensures that the incremental plastic dissipation is non-negative, and that ensures that the second law of thermodynamics is satisfied.

For general case where the material does exhibit cyclic softening, plastic free energy density $E_P$ is graphically described as the areas $adoc$ and $beoc$ at states $a$ and $b$, respectively. The formulation for plastic free energy density $E_P$ representing this assumption is given by:

$$E_P = \frac{1}{2} \left[ \sigma \left( \epsilon - \frac{\sigma}{E_0} - \epsilon_r \right) + \sigma_r \epsilon \right]$$  (507.12)

The incremental form of Equation 507.12 is:

$$dE_P = \frac{1}{2} \left[ (\sigma + \sigma_r)de + \left( \epsilon - \frac{1}{E_0} \sigma - \epsilon_r \right) d\sigma \right]$$  (507.13)

Adding Equation 507.10, 507.11, and 507.13, the incremental form of energy balance is achieved:

$$dE_S + dE_P + dD_P = \sigma d\epsilon$$  (507.14)

where the increment of three energy components add up to the increment of stress power during any loading step.

507.2.5 Modeling and Simulation Results

In this chapter, the simulation results are presented and compared with corresponding experimental results. Three sets of FEM simulations (Reg A, Reg B, and ASR A1) using the Real-ESSI Simulator system (Jeremić et al., 1988-2021) (http://real-essi.info/) are performed. In the following sections, the force–displacement responses of these three sets of simulations are shown and discussed. The strain and stress distributions, that share similar pattern in all three simulations, are also presented. In order to investigate the level of damage in the shear wall, the evolution of concrete damage index as well as the plastic dissipation density are plotted and discussed.

507.2.5.1 Force–Displacement Response

Figures 507.5 and 507.6 present the force–displacement responses of the simulation and experiment results for regular concrete (Reg A and Reg B) and ASR concrete.

The ESSI simulation curves show good matching of experimental results. The differences in the envelopes of the cyclic loading curves, between the numerical and experimental results are within 10%.
The shear strengths and failure loads/displacements given by ESSI simulations match well with the values determined by physical experiments. Note that this particular case with ASR concrete has a much larger unloading-reloading cyclic area, which means that ASR concrete has the capability of dissipating more input energy. It is important to note that this conclusion does not hold for other ASR concrete walls that were tested. This might indicate that for some structures with the ASR concrete, it is possible to dissipate more seismic energy if the structure is under earthquake cyclic loading. On the other hand, for some other structure with the ASR concrete, such conclusion might not hold as other test data suggests reduction of seismic energy dissipation capacity. This leads to the conclusion that variability of ASR concrete quality and material behavior can be significant.
507.2.5.2 Strain and Stress Distribution

Due to the fact that the stress, strain, and damage patterns in all three cases are very similar, only plots for the Reg A case is presented and discussed in the following sections.

Figure 507.7 shows the distribution of displacement components in the shear wall model at $u_y = 6$ mm. The vertical displacement distribution is almost symmetric. The left part of the shear wall is compressed down while the right part is extended up. It is noted that the upper support beam is thus not remaining horizontal, rather, it is applying bending load to the shear wall. The horizontal displacement in the top beam slab is almost uniform.

Figure 507.8 shows the distribution of strain components in the shear wall model at $u_y = 6$ mm. The dominant components of the strains are the vertical normal strain $\epsilon_{xx}$ due to the vertical loading, and the shear strain $\epsilon_{xy}$ due to the horizontal loading. As can be seen in the distribution of $\epsilon_{xx}$, the maximum tensile strain is much larger than the maximum compressive strain. Tensile strain actually means that there is a crack that opened at the location. This is expected since the compressive strength of concrete is higher than the tensile strength.

A $45^\circ$ shear zone can be observed in the shear wall, with significant amounts of tensile and shear strains. In addition, large tensile strains are developing around the top-left and bottom-right corners. As will be shown in the following section, the concrete in these areas is significantly damaged during the test.

Figure 507.9 shows the distribution of stress components in the shear wall model at $u_y = 6$ mm.
Figure 507.8: Distribution of strain components in the shear wall model at $u_y = 6$ mm: (a) Vertical normal strain $\varepsilon_{xx}$; (b) Horizontal normal strain $\varepsilon_{yy}$; (c) Shear strain $\varepsilon_{xy}$; (d) Strain magnitude $|\varepsilon|$. 

To make the plots more clear, only stresses in the shear wall is plotted. Again, a $45^\circ$ shear zone can be observed, that is consistent with the pattern in the strain plots.

Large compressive normal stresses are developed around the top-right and bottom-left corners, while large tensile normal stresses are observed around the top-left and bottom-right corners. The shear stresses at all four corners are significant.

Stress/force and strain/deformation distributions are not enough to directly quantify the level of damage in structures, especially in the case of cyclic loading. In order to analyze damage conditions, the concrete damage index and plastic energy dissipation are calculated and plotted, as shown in the next section.
Figure 507.9: Distribution of stress components in the shear wall model at $u_y = 6$ mm: (a) Vertical normal stress $\sigma_{xx}$; (b) Horizontal normal stress $\sigma_{yy}$; (c) Shear stress $\sigma_{xy}$; (d) Stress magnitude $|\sigma|$.

507.2.5.3 Concrete Damage and Energy Dissipation

Figure 507.10 shows the evolution of shear wall concrete damage index $d^+$, defined in Equation 104.598, at different levels of deformation during cyclic loading. Only tensile damage index is presented in Figure 507.10 as no compressive damage occurs in the wall. It is noted that, according to damage evolution from Figure 507.10, three damage/cracks zones are observed in the wall for load/displacement cycles of $u_y = \pm1.4$mm. Two $45^\circ$ tensile/shear zones with opposite directions as well as a tensile failure zone along the bottom of the wall are developing. Note that the flanges experience damage at the bottom due to tension, which indicates that wall-flange system experiences bending, and not pure shear as intended.
As the loading/deformation cycles increase, the damage zones extend and the level of damage increases. At the end, the entire wall is damaged. In reality, due to variability of material properties, some wall regions will experience localization of deformation, as observed in experiments. Modeling of localization of deformation suffers from mesh dependency effects. It is noted that localization of deformation zones, as seen in Figures 507.10 and 507.11 are about the size of single finite element. It is likely that upon mesh refinement these zones would change (Lu et al., 2009), however this type of sensitivity study was not done here due to time constraints. We also note that a more sound approach to modeling localized deformation behavior would be through the use of Cosserat continuum (Cosserat, 1909; de Borst, 1987). Cosserat continuum finite element and micropolar elastic and elastic-plastic
models are available within the Real-ESSI Simulator, however calibration of material parameters for those models from current test results is beyond the scope and time frame of this study, and was not done.

Other than damage index, plastic energy dissipation density can also be used to illustrate damage in structures. The main difference between damage index and plastic energy dissipation is that the plastic energy dissipation increases even after an element is completely damaged, that is when $d^+ = 1.0$.  

Figure 507.11 shows the evolution of plastic energy dissipation density in the shear wall model at different stages of loading. At the beginning of the test, Figure 507.11(a), the distribution of plastic energy dissipation density is very similar to the distribution of a damage index ($d^+$). However, as the load cycles progress and displacement increase, Figure 507.11(b), significant amount of energy is dissipated at bottom corners of both flanges. Most of that plastic dissipation happens due to opening of gaps, tensile cracking, and some shear. Plastic dissipation due to tensile cracking can happen only once, as cracks will not "heal", hence there is a redistribution of loads and deformation within flange-wall system. Once flange cannot dissipate any more energy, only wall is left to pick up loads and dissipate energy due to tension and shear. It is noted again that there was no observed development of compression damage, hence there is no plastic dissipation in compressed concrete. Toward the end of simulation, Figure 507.11(c), a failure zone that is X shaped, with distinct 45° tensile plastic dissipation zones in the wall, and the plastic dissipation zone at the corners, bottom and top of flanges, is observed. At the end of the simulation, Figure 507.11(d), a failure zone due to tension and shear, encompasses flange corners and most of the wall.

Figure 507.12 shows animation of plastic dissipation development in the wall/membrane.

---

2 Plastic dissipation is also very important for following seismic energy as it propagates through the soil structure system during earthquakes (Sinha et al., 2017; Wang et al., 2017). If seismic energy, and plastic dissipation can be accurately followed and its path even directed, during earthquake soil structure interaction (ESSI), soil structure systems can be optimized for safety and economy.
Figure 507.11: Evolution of plastic energy dissipation density in the shear wall model: (a) At $u_y = \pm 1.4\text{mm}$; (b) At $u_y = \pm 1.8\text{mm}$; (c) At $u_y = \pm 3.0\text{mm}$; (d) At $u_y = \pm 5.0\text{mm}$.
Figure 507.12: Animation of plastic dissipation in the wall/membrane.
Chapter 508

ESSI for Concrete Dams

(2019-2020-2021-)

(In collaboration with Dr. Han Yang and Dr. Hexiang Wang)
508.1 Chapter Summary and Highlights

508.2 Pine Flat Dam
Chapter 509

ESSI for Buildings

(In collaboration with Dr. Han Yang, and Dr. Hexiang Wang)
509.1 2D Frame with Energy Dissipation

Figure 509.1: 2D frame model.

Figure 509.2: Energy Dissipation for a frame on spread foundations.
Figure 509.3: Energy Dissipation for a frame on slab foundation.

Finite element model, input files for the Real-ESSI simulator for Concrete Frame model are available:

- All model files, one by one.
- Model archive (tar xz)

For uncompressed and un-taring the model archive file, please use

tar -xvf Concrete_frame_PEER_model.txz.
509.2 Ventura Hotel

509.2.1 Finite Element Model

Figure 509.4: Ventura hotel model.

Figure 509.5: Ventura hotel model.
Figure 509.6: Ventura hotel model.

Figure 509.7: Ventura hotel model.
Figure 509.8: Ventura hotel model, view in X direction.

Figure 509.9: Ventura hotel model, view in Y direction.
Figure 509.10: Ventura hotel model, view in Z direction.

Figure 509.11: Ventura hotel model, view along X direction.
Figure 509.12: Ventura hotel model, view along Y direction.

Figure 509.13: Ventura hotel model, view along Z direction.
Finite element model, input file for the Real-ESSI simulator for Ventura hotel model with and without SSI are available:

- Full SSI model, (7.8MB)
- Structure only, non-SSI model, (84kB)

**NOTE:** while full SSI model is only 11MB in compressed format, using `xz` compressor, fully uncompressed model files are over 3.7GB large, that is 3,770MB, or 3,868,356Kb.

For uncompressing and un-taring, please use

```
tar -xvf _Ventura_hotel_Full_SSI_Model_.tar.xz,
```
and

```
tar -xvf _Ventura_hotel_non_SSI_Model_.tar.xz.
```
509.3 Loma Linda Hospital

509.3.1 Finite Element Model

Figure 509.14: Loma Linda hospital 3D view.

Figure 509.15: Loma Linda hospital view in X direction.
Figure 509.16: Loma Linda hospital view in Y direction.

Figure 509.17: Loma Linda hospital view in Z direction.
509.4 ASCE-7 Model, Low, Steel Building

509.4.1 Finite Element Model

Figure 509.18: Low steel building model, 3D view.

Figure 509.19: Low steel building model, 3D view.
Figure 509.20: Low steel building model, 3D view.

Figure 509.21: Low steel building model, 3D view.
Figure 509.22: Low steel building model, 3D view.

Figure 509.23: Low steel building model, 3D view.
Finite element model, input file for the Real-ESSI simulator for full SSI ASCE-7 model, low, steel building are available HERE.

**NOTE:** This model is 1.1GB (1,102MB, 1,127,928Kb) in size even when compressed, due to large file describing incoherent motions. When uncompressed (use `tar -xvf _ASCE-7_low_steel_building_Model_.tar.xz`), models files grow to 3.0GB, that is 2,977MB, or 3,047,532Kb, so please be aware of disk space requirements for the model alone.
509.5 ASCE-7 Model, High, Concrete Building

509.5.1 Finite Element Model

Figure 509.24: High concrete building model, 3D view.
Figure 509.25: High concrete building model, XZ plane view.

Figure 509.26: High concrete building model, YZ plane view.
Figure 509.27: High concrete building model, cut through model view.

Figure 509.28: High concrete building model, cut through model view.
Finite element model, input file for the Real-ESSI simulator for full SSI ASCE-7 model, high concrete building are available HERE.

NOTE: This model is only 11MB large when compressed. When uncompressed (use tar -xvf _ASCE-7_tall_concrete_building_Model_.tar.xz), models files grow to 1.9GB, that is 1,867MB, or 1,911,524Kb, so please be aware of disk space requirements for the model alone.
Chapter 510


510.1 Motivation: Modeling and Simulation of Earthquake Soil Structure Interaction

Main motivation of this write-up (chapter, guidebook) is to provide a clear, practical, up to date guide on how to perform linear elastic and nonlinear, inelastic Earthquake Soil Structure Interaction (ESSI) modeling and simulations for infrastructure objects, including buildings, dams, bridges, nuclear installations, etc. This is particularly important at this time as a number of endeavors are underway to perform realistic ESSI analysis for a number of important soil, rock – structure systems, including dams, nuclear installations, bridges, buildings, etc.

This write-up is further motivated by the modeling and simulation challenges that are part of any soil, rock – structure system. These challenges are illustrated in Figure 510.1 for a number of soil, rock and soil/rock-structure systems.
Figure 510.1: ESSI modeling and simulation challenges: Free field motions, 3C/6C vs 3×1C; Nuclear Power Plant structure – soil/rock system, Small Modular Reactor structure – soil/rock system; Low and High Building-foundation-soil system; Dam-Foundation-Fluid system; Bridge-soil system. Aspects of modeling: 1) Seismic motions, 2) Inelastic soil and rock, 3) Inelastic interface/contact/joints, foundation with soil/rock and interfaces/contacts/joints within structure, 4) Inelastic structure, systems and components, 5) Solid, Structure – Fluid interaction, external (reservoirs, fluid pools...) and internal (fully saturated and partially, (un-)saturated soil, rock and concrete).
Presented are challenges related to the modeling of mechanics of ESSI problems.

1. Seismic motions: use of 1C, 3×1C and 3C motions, seismic motions input and radiation damping.

2. Inelastic, elastic plastic modeling of soil and rock, dry and/or partially or fully saturated, and energy dissipation in those soil/rock – structure components,

3. Inelastic, elastic plastic modeling of foundation concrete – soil/rock contacts/interfaces/joints that may be dry and/or partially or fully saturated, and energy dissipation in those parts of soil/rock – structure system,

4. Inelastic, elastic damage plastic modeling of structure, systems and components (SSCs). SSCs: beams, walls, plates, shell made of steel and reinforced concrete, base isolators and dissipators, systems and etc. and energy dissipation in those soil/rock – structure components,

5. Interaction of soil/rock – structure systems with internal, within structure or in pores of porous materials (soil, rock, concrete) and external fluids, reservoirs, pools, etc.

Presented are challenges related to the numerical simulation of ESSI problems.

A. Inelastic simulations on constitutive level, stress-strain, constitutive problem solutions

B. Inelastic simulations on finite element level, nonlinear system of equations solutions

C. Time marching algorithms, numerical damping

D. High performance, parallel computing
510.2 Introduction

Focus is on modeling and simulation of linear elastic and nonlinear, inelastic, elastic-plastic behavior of soil/rock – structure systems during earthquakes.

It is assumed that earthquake motions, earthquake field is known. Earthquake motion or earthquake field, can be given as a simple 1C (1 Component) vertically propagating shear wave, that is obtained from a de-convolution of a given (1C) surface motion, using, for example SHAKE type analysis. Earthquake motion or earthquake field can also be given as a full 3C (3 Component) wave field that is obtained from analytic wave propagation solutions or from a regional geophysical model simulations, using, for example SW4 type analysis. In addition, earthquake motions can be defined in a probabilistic way...
510.3 Seismic Energy Input and Dissipation

Recorded lectures, together with slides, about these topics are available at http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

More details about these topics are given in Section 109.2, on page 516 in Jeremić et al. (1989-2021).

510.3.1 Seismic Energy Input

Seismic energy input flux

510.3.2 Seismic Energy Dissipation

510.3.2.1 Seismic Energy Dissipation, Wave Reflection and Wave Radiation

Wave reflection and radiation damping

510.3.2.2 Seismic Energy Dissipation, Viscous Coupling

velocity proportional, viscous damping

510.3.2.3 Seismic Energy Dissipation, Material Inelasticity

Elastic-plastic energy dissipation of material, Displacement proportional

510.3.2.4 Seismic Energy Dissipation, Numerical, Algorithmic Positive and Negative Damping
510.4 Modeling: Seismic Motions

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

More details about these topics are given in Section 502.2, on page 2199 in Jeremić et al. (1989-2021).

510.4.1 Seismic Motions: Available Data

Details are given in Section 502.2.1, on page 2201 in Jeremić et al. (1989-2021).

510.4.2 Seismic Motion Development

Details are given in Section 502.2.3, on page 2205 in Jeremić et al. (1989-2021).

510.4.2.1 Seismic Motions from Empirical Models

Details are given in Section 109.2.3, on page 522 in Jeremić et al. (1989-2021).

510.4.2.2 Seismic Motions from Geophysical Models

Details are given in Section 109.2.6, on page 530 in Jeremić et al. (1989-2021).

Small Scale Geophysical Models.

Large Scale Regional Geophysical Models

510.4.2.3 Seismic Motions from 3D/3C Analytic Models

Details are given in Section 109.2.5, on page 526 in Jeremić et al. (1989-2021).

510.4.2.4 Seismic Motions from Full Waveform Inversion

This is based on recent work by Guidio and Jeong (2019); Guidio et al. (2020).

510.4.3 6C vs 3C vs 3×1C vs 1C Seismic Motions

Details are given in Section 504.4.1, on page 2442 in Jeremić et al. (1989-2021).
510.4.4 Incoherent Seismic Motions

Details are given in Section 109.2.8, on page 531 in Jeremić et al. (1989-2021).

510.4.5 Seismic Motion Input into FEM Models

Details are given in Section 109.4.1, on page 537 in Jeremić et al. (1989-2021).
510.5  Modeling: Inelastic, Nonlinear Material Modeling for Solids and Structures

Recorded lectures, together with slides, about these topics are available at http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

510.5.1  Inelastic Material Modeling of Rock

Details are given in Section 104.8, on page 309 in Jeremić et al. (1989-2021).

510.5.1.1  Calibration of Inelastic Material Model Parameters for Rock

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).

510.5.2  Inelastic Material Modeling of Soil

510.5.2.1  Dry Soil

Details are given in Section 104.6, on page 207 in Jeremić et al. (1989-2021).

510.5.2.2  Fully Saturated Soil

Details are given in Section 102.12, on page 131 in Jeremić et al. (1989-2021).

510.5.2.3  Partially Saturated, Unsaturated Soil

Details are given in Section 102.12.2, on page 146 in Jeremić et al. (1989-2021).

510.5.2.4  Calibration of Inelastic Material Model Parameters for Soil

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).

510.5.3  Inelastic Material Modeling of Steel

Details are given in Section 104.9, on page 343 in Jeremić et al. (1989-2021).

510.5.3.1  Calibration of Inelastic Material Model Parameters for Steel

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).
510.5.4 Inelastic Material Modeling of Concrete

Solids, Beams, Plates, Walls and Shells

Details are given in Section 104.9, on page 343 in Jeremić et al. (1989-2021).

510.5.4.1 Calibration of Inelastic Material Model Parameters for Concrete

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).
510.6 Modeling: Inelastic, Nonlinear Material Modeling for Contacts, Interfaces, and Joints

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

510.6.1 Material Modeling of Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

Details are given in Section 104.7, on page 286 in Jeremić et al. (1989-2021).

510.6.1.1 Calibration of Inelastic Material Model Parameters for Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

510.6.2 Material Modeling of Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

Details are given in Section 104.7, on page 286 in Jeremić et al. (1989-2021).

510.6.2.1 Calibration of Inelastic Material Model Parameters for Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

510.7 Modeling: Buoyancy

Details are given in Section 502.3.5, on page 2287 in Jeremić et al. (1989-2021).
510.8 Modeling: Base Isolator and Base Dissipator Systems

510.8.1 Base Isolator Systems

Details are given in Section 102.11, on page 130, in Jeremić et al. (1989-2021).

510.8.2 Base Dissipator Systems

Details are given in Section 102.11, on page 130, in Jeremić et al. (1989-2021).

510.8.2.1 Calibration of Elastic/Inelastic Material Model Parameters for Base Dissipator Systems
510.9 Modelling: Finite Element System

510.9.1 Mass Matrix

Details are given in Section 102.4, on page 108, Section 102.6, on page 120, Section 102.7, on page 120, Section 102.8, on page 123, Section 102.9, on page 129, Section 102.10, on page 129, Section 102.11, on page 130, and Section 102.12, on page 131, in Jeremić et al. (1989-2021).

510.9.1.1 Consistent Mass Matrix

510.9.1.2 Lumped Mass Matrix

510.9.2 Viscous Damping Matrix

Details are given in Section 108.4, on page 509, in Jeremić et al. (1989-2021).

510.9.2.1 Rayleigh Damping

510.9.2.2 Caughey Damping

510.9.3 Stiffness Matrix

Details are given in Section 102.4, on page 108, Section 102.6, on page 120, Section 102.7, on page 120, Section 102.8, on page 123, Section 102.9, on page 129, Section 102.10, on page 129, Section 102.11, on page 130, and Section 102.12, on page 131, in Jeremić et al. (1989-2021).

510.9.3.1 Tangent Stiffness Matrix

510.9.3.2 Consistent Stiffness Matrix
510.10 Modeling: Solid, Structure – Fluid Interaction Modeling

Details of OpenFOAM – Real-ESSI Simulator coupling are available starting with Section 111.2, on page 653, in Jeremić et al. (1989-2021).
510.11 Simulation: Nonlinear Finite Elements

Details are given in Section 102.2, on page 93 in Jeremić et al. (1989-2021).

510.11.1 Time Marching Algorithms for Solution of Nonlinear Equations of Motion

Details are given in Section 108.3, on page 507 in Jeremić et al. (1989-2021).

510.11.1.1 Newmark Algorithm

Details are given in Section 108.3.1, on page 507 in Jeremić et al. (1989-2021).

510.11.1.2 Hilber Hughes Taylor $\alpha$ Algorithm

Details are given in Section 108.3.2, on page 508 in Jeremić et al. (1989-2021).

510.11.2 Solution of Elastic-Plastic Constitutive Equations

Details are given in Section 104.2.2, on page 167 in Jeremić et al. (1989-2021).

510.11.2.1 Explicit Integration of Elastic-Plastic Constitutive Equations

Details are given in Section 104.3, on page 192 in Jeremić et al. (1989-2021).

  Error accumulation.

510.11.2.2 Implicit Integration of Elastic-Plastic Constitutive Equations

Details are given in Section 104.4, on page 193 in Jeremić et al. (1989-2021).

  Iterations and tolerance issues.
510.12 Modelling Guide for ESSI

Recorded lectures, together with slides, about these topics are available at http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

More details about these topics are given in Section 502.3, on page 2271 in Jeremić et al. (1989-2021).

510.12.1 Buildings and NPPs on Shallow Foundations, Models

Details are given in Section 504.6, on page 2452 in Jeremić et al. (1989-2021).

510.12.2 Buildings and NPPs on Deeply Embedded Foundation (SMRs), Models

Details are given in Section 504.7, on page 2466 in Jeremić et al. (1989-2021).

510.12.3 Buildings and NPPs on Piles and Pile Group Foundations, Models

Details are given in Section 504.6, on page 2452 in Jeremić et al. (1989-2021).

510.12.4 Structure – Soil – Structure Interaction, Models

Details are given in Section 504.8, on page 2482 in Jeremić et al. (1989-2021).
510.13 Practical Steps for Inelastic ESSI Analysis

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

See Model Development section in Pecker et al. (2021).

510.13.1 Model Development for ESSI

Details are given in Section 502.4, on page 2293 in Jeremić et al. (1989-2021).

510.13.2 Earthquake Soil Structure Interaction: Model Analysis

510.13.3 Earthquake Soil Structure Interaction: Results Postprocessing

Details are given in Section 208.2, on page 1245 in Jeremić et al. (1989-2021).
510.14 Quality Assurance Procedures for ESSI Modeling and Simulation

Details of the quality assurance are given in Sections 313.3, on page 1828 and 313.4, on page 1831, in Jeremić et al. (1989-2021).

Moreover, verification procedures for ESSI modeling is given in part 300, on page 1380 in Jeremić et al. (1989-2021).

See also Verification and Validation section in Pecker et al. (2021).

510.14.1 Verification

510.14.2 Validation
510.15 Practical Examples, Nonlinear, Inelastic ESSI

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

510.15.1 Nuclear Power Plant, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

510.15.2 Nuclear Power Plant on Piles, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

510.15.3 Nuclear Power Plant, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 6C/3C/3×1C/1C Seismic Motions

510.15.4 Small Modular Reactor, Deeply Embedded, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

510.15.5 Small Modular Reactor, Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil (Cyclic Mobility and Liquefaction), Inelastic Saturated Contact/Interface (Buoyant Pressures), 6C/3C/3×1C/1C Seismic Motions

510.15.6 Multiple Buildings and Nuclear Power Plants (Structure-Soil-Structure Interaction), Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

510.15.7 Multiple Small Modular Reactors (Structure-Soil-Structure Interaction), Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 3C Seismic Motions
Chapter 511

ASCE-4, Chapter on Nonlinear ESSI analysis

(2016-2020-2021-)

2589
511.1 Motivation: Modeling and Simulation of Earthquake Soil Structure Interaction

Main motivation of this write-up, chapter is to provide a clear, practical, up to date guide on how to perform nonlinear, inelastic Earthquake Soil Structure Interaction (ESSI) modeling and simulations for nuclear installations. This is particularly important at this time as a number of endeavors are underway to perform realistic ESSI analysis for a number of nuclear installations.

This write-up is further motivated by the modeling and simulation challenges that are part of any soil, rock – structure system. These challenges are illustrated in Figure 511.1 for a number of soil, rock and soil/rock-structure systems.
Figure 511.1: ESSI modeling and simulation challenges: Free field motions, 3C vs 3×1C; Nuclear Power Plant structure – soil/rock system, Small Modular Reactor structure – soil/rock system. Aspects of modeling: 1) Seismic motions, 2) Inelastic soil and rock, 3) Inelastic interface/contact/joints, foundation with soil/rock and interfaces/contacts/joints within structure, 4) Inelastic structure, systems and components, 5) Interaction with external (reservoirs, fluid pools...) and internal (saturated and un-saturated soil/rock and concrete.)
Presented are challenges related to the modeling of mechanics of ESSI problems.

1. Seismic motions: use of 1C, 3×1C and 3C motions, seismic motions input and radiation damping.

2. Inelastic, elastic plastic modeling of soil and rock, dry and/or partially or fully saturated, and energy dissipation in those soil/rock – structure components,

3. Inelastic, elastic plastic modeling of foundation concrete – soil/rock contacts/interfaces/joints that may be dry and/or partially or fully saturated, and energy dissipation in those parts of soil/rock – structure system,

4. Inelastic, elastic damage plastic modeling of structure, systems and components (SSCs). SSCs: beams, walls, plates, shell made of steel and reinforced concrete, base isolators and dissipators, systems and etc. and energy dissipation in those soil/rock – structure components,

5. Interaction of soil/rock – structure systems with internal, within structure or in pores of porous materials (soil, rock, concrete) and external fluids, reservoirs, pools, etc.

Presented are challenges related to the numerical simulation of ESSI problems.

A. Inelastic simulations on constitutive level, stress-strain, constitutive problem solutions

B. Inelastic simulations on finite element level, nonlinear system of equations solutions

C. Time marching algorithms, numerical damping

D. High performance, parallel computing
511.2 Introduction

Focus is on modeling and simulation of linear elastic and nonlinear, inelastic, elastic-plastic behavior of soil/rock – structure systems during earthquakes.

It is assumed that earthquake motions, earthquake field is known. Earthquake motion or earthquake field, can be given as a simple 1C (1 Component) vertically propagating shear wave, that is obtained from a de-convolution of a given (1C) surface motion, using, for example SHAKE type analysis. Earthquake motion or earthquake field can also be given as a full 3C (3 Component) wave field that is obtained from analytic wave propagation solutions or from a regional geophysical model simulations, using, for example SW4 type analysis. In addition, earthquake motions can be defined in a probabilistic way...
511.3 Seismic Energy Input and Dissipation

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

More details about these topics are given in Section 109.2, on page 516 in Jeremić et al. (1989-2021).

511.3.1 Seismic Energy Input

Seismic energy input flux

511.3.2 Seismic Energy Dissipation

511.3.2.1 Seismic Energy Dissipation, Wave Reflection and Wave Radiation

Wave reflection and radiation damping

511.3.2.2 Seismic Energy Dissipation, Viscous Coupling

Velocity proportional, viscous damping

511.3.2.3 Seismic Energy Dissipation, Material Inelasticity

Elastic-plastic energy dissipation of material, Displacement proportional

511.3.2.4 Seismic Energy Dissipation, Numerical, Algorithmic Positive and Negative Damping
511.4 Modeling: Seismic Motions

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

More details about these topics are given in Section 502.2, on page 2199 in Jeremić et al. (1989-2021).

511.4.1 Seismic Motions: Available Data

Details are given in Section 502.2.1, on page 2201 in Jeremić et al. (1989-2021).

511.4.2 Seismic Motion Development

Details are given in Section 502.2.3, on page 2205 in Jeremić et al. (1989-2021).

511.4.2.1 Seismic Motions from Empirical Models

Details are given in Section 109.2.3, on page 522 in Jeremić et al. (1989-2021).

511.4.2.2 Seismic Motions from Geophysical Models

Details are given in Section 109.2.6, on page 530 in Jeremić et al. (1989-2021).

Small Scale Geophysical Models.

Large Scale Regional Geophysical Models

511.4.2.3 Seismic Motions from 3D/3C Analytic Models

Details are given in Section 109.2.5, on page 526 in Jeremić et al. (1989-2021).

511.4.3 6C vs 3C vs 3×1C vs 1C Seismic Motions

Details are given in Section 504.4.1, on page 2442 in Jeremić et al. (1989-2021).

511.4.4 Incoherent Seismic Motions

Details are given in Section 109.2.8, on page 531 in Jeremić et al. (1989-2021).
511.4.5 Seismic Motion Input into FEM Models

Details are given in Section 109.4.1, on page 537 in Jeremić et al. (1989-2021).
511.5 Modeling: Inelastic, Nonlinear Material Modeling for Solids and Structures

Recorded lectures, together with slides, about these topics are available at http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

511.5.1 Inelastic Material Modeling of Rock

Details are given in Section 104.8, on page 309 in Jeremić et al. (1989-2021).

511.5.1.1 Calibration of Inelastic Material Model Parameters for Rock

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).

511.5.2 Inelastic Material Modeling of Soil

511.5.2.1 Dry Soil

Details are given in Section 104.6, on page 207 in Jeremić et al. (1989-2021).

511.5.2.2 Fully Saturated Soil

Details are given in Section 102.12, on page 131 in Jeremić et al. (1989-2021).

511.5.2.3 Partially Saturated, Unsaturated Soil

Details are given in Section 102.12.2, on page 146 in Jeremić et al. (1989-2021).

511.5.2.4 Calibration of Inelastic Material Model Parameters for Soil

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).

511.5.3 Inelastic Material Modeling of Steel

Details are given in Section 104.9, on page 343 in Jeremić et al. (1989-2021).

511.5.3.1 Calibration of Inelastic Material Model Parameters for Steel

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).
511.5.4 Inelastic Material Modeling of Concrete

Solids, Beams, Plates, Walls and Shells

Details are given in Section 104.9, on page 343 in Jeremić et al. (1989-2021).

511.5.4.1 Calibration of Inelastic Material Model Parameters for Concrete

Details are given in Section 104.10, on page 348 in Jeremić et al. (1989-2021).
511.6 Modeling: Inelastic, Nonlinear Material Modeling for Contacts, Interfaces, and Joints

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

511.6.1 Material Modeling of Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

Details are given in Section 104.7, on page 286 in Jeremić et al. (1989-2021).

511.6.1.1 Calibration of Inelastic Material Model Parameters for Dry Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

511.6.2 Material Modeling of Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

Details are given in Section 104.7, on page 286 in Jeremić et al. (1989-2021).

511.6.2.1 Calibration of Inelastic Material Model Parameters for Saturated Contacts, Interfaces, and Joints (Concrete, Steel – Soil, Rock)

511.7 Modeling: Buoyancy

Details are given in Section 502.3.5, on page 2287 in Jeremić et al. (1989-2021).
511.8  Modeling: Base Isolator and Base Dissipator Systems

511.8.1  Base Isolator Systems

Details are given in Section 102.11, on page 130, in Jeremić et al. (1989-2021).

511.8.2  Base Dissipator Systems

Details are given in Section 102.11, on page 130, in Jeremić et al. (1989-2021).

511.8.2.1  Calibration of Elastic/Inelastic Material Model Parameters for Base Dissipator Systems
511.9 Modelling: Finite Element System

511.9.1 Mass Matrix

Details are given in Section 102.4, on page 108, Section 102.6, on page 120, Section 102.7, on page 120, Section 102.8, on page 123, Section 102.9, on page 129, Section 102.10, on page 129, Section 102.11, on page 130, and Section 102.12, on page 131, in Jeremić et al. (1989-2021).

511.9.1.1 Consistent Mass Matrix

511.9.1.2 Lumped Mass Matrix

511.9.2 Viscous Damping Matrix

Details are given in Section 108.4, on page 509, in Jeremić et al. (1989-2021).

511.9.2.1 Rayleigh Damping

511.9.2.2 Caughey Damping

511.9.3 Stiffness Matrix

Details are given in Section 102.4, on page 108, Section 102.6, on page 120, Section 102.7, on page 120, Section 102.8, on page 123, Section 102.9, on page 129, Section 102.10, on page 129, Section 102.11, on page 130, and Section 102.12, on page 131, in Jeremić et al. (1989-2021).

511.9.3.1 Tangent Stiffness Matrix

511.9.3.2 Consistent Stiffness Matrix
511.10 Modeling: Solid, Structure – Fluid Interaction Modeling

Details of OpenFOAM – Real-ESSI Simulator coupling are available starting with Section 111.2, on page 653, in Jeremić et al. (1989-2021).
511.11 Simulation: Nonlinear Finite Elements

Details are given in Section 102.2, on page 93 in Jeremić et al. (1989-2021).

511.11.1 Time Marching Algorithms for Solution of Nonlinear Equations of Motion

Details are given in Section 108.3, on page 507 in Jeremić et al. (1989-2021).

511.11.1.1 Newmark Algorithm

Details are given in Section 108.3.1, on page 507 in Jeremić et al. (1989-2021).

511.11.1.2 Hilber Hughes Taylor $\alpha$ Algorithm

Details are given in Section 108.3.2, on page 508 in Jeremić et al. (1989-2021).

511.11.2 Solution of Elastic-Plastic Constitutive Equations

Details are given in Section 104.2.2, on page 167 in Jeremić et al. (1989-2021).

511.11.2.1 Explicit Integration of Elastic-Plastic Constitutive Equations

Details are given in Section 104.3, on page 192 in Jeremić et al. (1989-2021).
   Error accumulation.

511.11.2.2 Implicit Integration of Elastic-Plastic Constitutive Equations

Details are given in Section 104.4, on page 193 in Jeremić et al. (1989-2021).
   Iterations and tolerance issues.
511.12 Modelling Guide for ESSI

Recorded lectures, together with slides, about these topics are available at
http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

More details about these topics are given in Section 502.3, on page 2271 in Jeremić et al. (1989-2021).

511.12.1 Buildings and NPPs on Shallow Foundations, Models
Details are given in Section 504.6, on page 2452 in Jeremić et al. (1989-2021).

511.12.2 Buildings and NPPs on Deeply Embedded Foundation (SMRs), Models
Details are given in Section 504.7, on page 2466 in Jeremić et al. (1989-2021).

511.12.3 Buildings and NPPs on Piles and Pile Group Foundations, Models
Details are given in Section 504.6, on page 2452 in Jeremić et al. (1989-2021).

511.12.4 Structure – Soil – Structure Interaction, Models
Details are given in Section 504.8, on page 2482 in Jeremić et al. (1989-2021).
511.13 Practical Steps for Inelastic ESSI Analysis

Recorded lectures, together with slides, about these topics are available at http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

See Model Development section in Pecker et al. (2021).

511.13.1 Model Development for ESSI

Details are given in Section 502.4, on page 2293 in Jeremić et al. (1989-2021).

511.13.2 Earthquake Soil Structure Interaction: Model Analysis

511.13.3 Earthquake Soil Structure Interaction: Results Postprocessing

Details are given in Section 208.2, on page 1245 in Jeremić et al. (1989-2021).
511.14 Quality Assurance Procedures for ESSI Modeling and Simulation

Details of the quality assurance are given in Sections 313.3, on page 1828 and 313.4, on page 1831, in Jeremić et al. (1989-2021).

Moreover, verification procedures for ESSI modeling is given in part 300, on page 1380 in Jeremić et al. (1989-2021).

See also Verification and Validation section in Pecker et al. (2021).

511.14.1 Verification

511.14.2 Validation
511.15 Practical Examples, Nonlinear, Inelastic ESSI

Recorded lectures, together with slides, about these topics are available at

http://sokocalo.engr.ucdavis.edu/~jeremic/Real_ESSI_Simulator/OnlineLectures

511.15.1 Nuclear Power Plant, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.2 Nuclear Power Plant on Piles, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.3 Nuclear Power Plant, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 6C/3C/3×1C/1C Seismic Motions

511.15.4 Small Modular Reactor, Deeply Embedded, Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.5 Small Modular Reactor, Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil (Cyclic Mobility and Liquefaction), Inelastic Saturated Contact/Interface (Buoyant Pressures), 6C/3C/3×1C/1C Seismic Motions

511.15.6 Multiple Buildings and Nuclear Power Plants (Structure-Soil-Structure Interaction), Inelastic Structure, Inelastic Soil, Inelastic Contact/Interface, 6C/3C/3×1C/1C Seismic Motions

511.15.7 Multiple Small Modular Reactors (Structure-Soil-Structure Interaction), Deeply Embedded, High Water Table, Inelastic Structure, Inelastic Soil, Cyclic Mobility and Liquefaction, Inelastic Saturated Contact/Interface, Buoyant Pressures, 3C Seismic Motions
Chapter 512

Earthquake-Soil-Structure Interaction, Core Functionality

(In collaboration with Dr. Yuan Feng, Dr. Han Yang and Dr. Hexiang Wang)
512.1 Core Functionality for ESSI Analysis of Nuclear Installations

Presented here are models that represent core functionality for elastic and inelastic analysis of infrastructure objects, including nuclear installations. There exist a number of other models, with different sophistication levels, that can be used, depending on the amount of data available, about the soil, rock, concrete, contacts/interfaces and seismic motions (Jeremić et al., 1989-2021). However, in order to begin to use of inelastic/nonlinear analysis, and assess inelastic/nonlinear effects on a dynamic response of soil structure systems, a set of initial models and analysis parameters are needed. Provided below is a set of models and materials parameters that are recommended for initial use of inelastic/nonlinear analysis of soil structure systems, using the Real-ESSI Simulator system. (http://real-essi.info/). It is noted that a detailed description of examples, commands and the Real-ESSI Simulator system is provided by Jeremić et al. (1989-2021, 1988-2021) and is also available at the Real-ESSI Simulator website http://real-essi.info/. In addition, preprocessing, model development and postprocessing, results visualization for the Real-ESSI Simulator system is also described in detail pre and post processing documents that are available at http://real-essi.info/.

512.2 Model Setup

Each model has to be named:

    model name "model_name_string";

In addition to that, there are a number of other considerations to be aware of:

- Each command line has to end with a semicolon ";"
- Comment on a line begins with either "//" or "!" and last until the end of current line.
- Units are required (see more below) for all quantities and variables.
- Include statements allow splitting source into several files
- All variables are double precision (i.e. floats) with a unit attached.
- All standard arithmetic operations are implemented, and are unit sensitive.
- Internally, all units are represented in the base SI units (m - s - kg).
- The syntax ignores extra white spaces, tabulations and newlines. Wherever they appear, they are there for code readability only. (This is why all commands need to end with a semicolon).
512.3 Linear Elastic Modeling

for single stage linear elastic modeling, one stage of loading has to be defined

```
new loading stage "self weight loading stage";
```

512.4 Nonlinear/Inelastic Modeling

For inelastic modeling, stages of loading have to be defined in proper sequence.

```
new loading stage "self weight loading stage";
```

```
. . .
```

```
new loading stage "Seismic Loading";
```

```
. . .
```

512.5 Model Domain

Finite element model is developed by defining the finite element mesh which is made of nodes, finite elements, the material, and the loads.
512.5.1 Nodes

For example:

```
add node No 1 at (1.0*m, 2.5*m, 3.33*m) with 3 dofs;
```

adds a node number 1 at coordinates \( x = 1.0m, y = 2.5m \) and \( z = 3.33m \) with 3 dofs. The nodes can be of 3dofs \([u_x, u_y, u_z]\), 4dofs \([u_x, u_y, u_z, p]\) (u-p elements) , 6dofs \([u_x, u_y, u_z, r_x, r_y, r_z]\) (beams and shells) and 7 dofs \([u_x, u_y, u_z, p, U_x, U_y, U_z]\) (upU element) types.

512.5.2 Boundary Conditions

Example fix translation \( x \) and \( y \) for node #3 fix node # 3 dofs ux uy;

Example fix all appropriate DOFs for node #7. fix node # 7 dofs all;

512.5.3 Static Acceleration Field

Example adding acceleration induced loading field for (some) elements

```
add acceleration field # 1
ax = 0*m/s^2
ay = 0*m/s^2
az = -9.81*m/s^2;
```

512.5.4 Dynamic Acceleration Field, Earthquake

One Example of add DRM load from wave fields:

```
add load # 1 type DRM from wave field
  # 1 in direction ux
  # 2 in direction uy
soil_surface at z = 60.0*m
hdf5_file = "input.hdf5";
```

512.5.5 Super Element

Super element is defined by providing mass and stiffness matrix, together with nodes and DOF numbering. It is assumed that the Super Element is a linear elastic element that is made up of a number of other finite elements. Super Element represents a part of model (structure, solid) that is linear elastic, and that has stiffness and mass matrix already defined using other finite element programs. Other finite element programs export stiffness and mass matrix. In addition, information about Super Element node numbers and degrees of freedom (DOFs) needs to be supplied as well.
512.6 Structural Modeling

Presented in this section are models that are used for modeling and simulation of structural behavior. Following the usually made assumption that structural components will remain linear elastic, only linear elastic material is used for structural modeling. It is noted, that fully nonlinear (inelastic, elastic-damage-plastic) models are also available for modeling of structural components (Jeremić et al., 1989-2021, 1988-2021). However, for the purpose of presenting core functionality features, those models are not covered here.

It is noted that a complete structural model can be replaced with one linear elastic super element, as described in section 512.5.5 on page 2611.

512.6.1 Truss

Truss element represents a 3D two node linear geometry truss member. Real-ESSI command for truss element is given in detail in section ??.

```plaintext
1 add element # 1 type truss
2 with nodes (1,2)
3 use material # 1
4 cross_section = 1*mm^2
5 mass_density = 2000*kg/m^3;
```

512.6.2 Beam

Beam finite element represents a 3D linear geometry, two node Bernoulli beam member, with 6 DOFs per node. Real-ESSI command for beam element is given in detail in section ??.

```plaintext
1 add element # 1 type beam_elastic
2 with nodes (1, 2)
3 cross_section = 1*mm^2
4 elastic_modulus = 2e8*Pa
5 shear_modulus = 1e8*Pa
6 torsion_Jx = 0.33*mm^4
7 bending_Iy = 1.0/12*mm^4
8 bending_Iz = 1.0/12*mm^4
9 mass_density = 2000*kg/m^3
10 xz_plane_vector = (1, 0, 1)
11 joint_1_offset = (0*mm, 0*mm, 0*mm)
12 joint_2_offset = (0*mm, 0*mm, 0*mm);
```
512.6.3 Shell

Shell finite element represents a 3D linear elastic geometry, 4 node ANDES shell member with 6DOFs per node, including drilling DOFs (in plane twist). Real-ESSI command for shell element is given in detail in section ??.

```
1 add element # 1 type 4NodeShell_ANDES
2     with nodes (1,2,3,4)
3     use material # 1
4     thickness = 1*m ;
```

512.7 Solid Modeling

Presented in this section are models that are used for modeling and simulation of soils, using solid and contact/interface elements for interface of foundations and soil. Models for soil can be linear elastic, while they can also be nonlinear/inelastic, mimicking simple $G/G_{max}$ behavior. Models for contact/interface can represent bonded contact, where no slip or gapping is allowed, and also a frictional slip and gapping contact/interface.

It is noted, that a number of more or less sophisticated material models for soil and for contact/interface are also available (Jeremić et al., 1989-2021, 1988-2021). However, for the purpose of presenting core functionality features, those models are not covered here.

512.7.1 Solid Brick

Solid brick finite element with 8 nodes, linear interpolation of displacements between nodes, and three DOFs per node is available. This element is very good for modeling soil volume close to and far away from the structural. Real-ESSI command for 8 node solid brick is given in detail in section ??.

```
1 add element # 1 type 8NodeBrick
2          using 2 Gauss points each direction
3          with nodes (1, 2, 3, 4, 5, 6, 7, 8)
4          use material # 1;
```

512.7.2 Contact, Interfaces, Joints

```
1 add element # 1 type StressBasedSoftContact_NonLinHardShear
2          with nodes (1, 2)
3          initial_axial_stiffness = 5*MPa
4          stiffening_rate = 100
5          max_axial_stiffness = 800*MPa
```
512.8 Core Material Modeling Parameters for Soil, Rock, Concrete, and Steel

512.8.1 Linear and Nonlinear Elastic Soil, Rock, Concrete, and Steel Modeling

512.8.2 Inelastic/Nonlinear Soil Modeling

Simple modeling of soil can be done using the so called stiffness degradation curves, or \( G/G_{\text{max}} \) curves, and damping curves, as developed by Seed and Idriss (1970a).

As an example, an elastic plastic material model based on von Mises yield surface with isotropic hardening or softening and Armstrong Frederick nonlinear kinematic hardening can be used to develop such curves. Model parameters are given below:

```plaintext
add material # 1 type vonMisesArmstrongFrederick
  mass_density = 2500*kg/m^3
  elastic_modulus = 30 * MPa
  poisson_ratio = 0.3
  von_mises_radius = 300 * Pa
  armstrong_frederick_ha = 150 * MPa
  armstrong_frederick_cr = 25000
  isotropic_hardening_rate = 0*Pa;
```

while the corresponding \( G/G_{\text{max}} \) and damping curves are given in Figure 512.2.

It is noted that von Mises Armstrong-Frederick Nonlinear Kinematic Hardening material model is a full 3D elastic plastic material model, that is capable of modeling \( G/G_{\text{max}} \) and damping behavior, defined in 1D shear testing, fairly well in full 3D.

The command is

```plaintext
add material # <.> type vonMisesArmstrongFrederick
  mass_density = <M/L^3>
  elastic_modulus = <F/L^2>
  poisson_ratio = <.>
  von_mises_radius = <.>
  armstrong_frederick_ha = <F/L^2>
  armstrong_frederick_cr = <.>
  isotropic_hardening_rate = <F/L^2> ;
```
Figure 512.2: Stiffness degradation ($G/G_{\text{max}}$) and damping curves developed using von Mises Armstrong-Frederick Nonlinear Kinematic Hardening material model.

512.8.3 Inelastic/Nonlinear Rock Modeling

512.8.4 Inelastic/Nonlinear Concrete Modeling

512.8.5 Inelastic/Nonlinear Steel Modeling

512.9 Core Material Modeling Parameters for Contacts, Interfaces and Joints
The command for stress based dry soft nonlinear hardening is:

```
add element # 1 type StressBasedSoftContact_NonLinHardShear
   with nodes (1, 2)
   initial_axial_stiffness = 5*MPa
   stiffening_rate = 100
   max_axial_stiffness = 800*MPa
   initial_shear_stiffness = 800*kPa
   axial_viscous_damping = 50*Pa*s
   shear_viscous_damping = 50*Pa*s
   residual_friction_coefficient = 0.68
   shear_zone_thickness = 5e-3*m
   contact_plane_vector = (0, 0, 1 );
```

### 512.9.1 Mass Concrete Against Silt, Sand, Gravel and Clay

A set of initial recommended material parameters for frictional contact/interface are given in Tables 512.1 for contact between mass concrete and sand/gravel. Frictional properties given below are recommended by NAVFAC (1986).

Table 512.1: Friction coefficients for contact/interface of dissimilar materials, mass concrete against soil.

<table>
<thead>
<tr>
<th>Mass concrete on soil</th>
<th>Friction coefficient ((\tan \phi))</th>
<th>Friction angle ((\phi))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean sound rock</td>
<td>0.70</td>
<td>35(^{\circ})</td>
</tr>
<tr>
<td>Clean gravel, gravel sand mixture, coarse sand</td>
<td>0.55 – 0.60</td>
<td>29(^{\circ}) – 31(^{\circ})</td>
</tr>
<tr>
<td>Clean fine to medium sand, silty medium to coarse sand</td>
<td>0.45 – 0.55</td>
<td>24(^{\circ}) – 29(^{\circ})</td>
</tr>
<tr>
<td>Fine sandy silt, nonplastic silt</td>
<td>0.35 – 0.45</td>
<td>19(^{\circ}) – 24(^{\circ})</td>
</tr>
<tr>
<td>Very stiff clay</td>
<td>0.40 – 0.50</td>
<td>22(^{\circ}) – 27(^{\circ})</td>
</tr>
</tbody>
</table>

Example command for a contact/interface element for mass concrete against clean sand, silty sand-gravel mix, single size rock fill (friction coefficient 0.30) is given below:

```
add element # 1 type StressBasedSoftContact_NonLinHardShear
   with nodes (1, 2)
   initial_axial_stiffness = 10 * MPa
   stiffening_rate = 100
   max_axial_stiffness = 50 * MPa
   initial_shear_stiffness = 40 * kPa
   axial_viscous_damping = 100 * Pa * s
   shear_viscous_damping = 100 * Pa * s
   residual_friction_coefficient = 0.30
   shear_zone_thickness = 5e-3*m
```
Another example, for contact/interface element for mass concrete against clean gravel, gravel sand mixture, coarse sand (friction coefficient 0.55-0.60) is given below:

```
add element # 1 type StressBasedSoftContact_NonLinHardShear
with nodes ( 1, 2)
initial_axial_stiffness = 20 * MPa
stiffening_rate = 100
max_axial_stiffness = 100 * MPa
initial_shear_stiffness = 80 * kPa
axial_viscous_damping = 200 * Pa * s
shear_viscous_damping = 200 * Pa * s
residual_friction_coefficient = 0.55
shear_zone_thickness = 1e-2*m
contact_plane_vector = (0, 0, 1);
```

512.9.2 Steel Sheet Against Sand, Gravel and Rockfill

Recommended material parameters for frictional contact/interface of steel sheets against sand and gravel are given in Tables 512.2. Frictional properties given below are recommended by NAVFAC (1986).

Table 512.2: Friction coefficients for contact/interface of dissimilar materials, steel sheet piles against soil.

<table>
<thead>
<tr>
<th>Steel sheets against soil</th>
<th>Friction coefficient (tan φ)</th>
<th>Friction angle (φ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean gravel, sand-gravel mix, well graded rock fill</td>
<td>0.40</td>
<td>22°</td>
</tr>
<tr>
<td>Clean sand, silty sand-gravel mix, single size rock fill</td>
<td>0.30</td>
<td>17°</td>
</tr>
<tr>
<td>Fine sandy silt, nonplastic silt</td>
<td>0.20</td>
<td>11°</td>
</tr>
</tbody>
</table>

Example commands for contact/interface element for steel sheets against clean sand, silty sand-gravel mix, single size rock fill (friction coefficient 0.30) is given below:

```
add element # 1 type StressBasedSoftContact_NonLinHardShear
with nodes ( 1, 2)
initial_axial_stiffness = 1000 * MPa
stiffening_rate = 100
max_axial_stiffness = 5 * GPa
initial_shear_stiffness = 4 * MPa
axial_viscous_damping = 100 * Pa * s
shear_viscous_damping = 100 * Pa * s
residual_friction_coefficient = 0.30
```
and for steel sheets against clean gravel, sand-gravel mix, well graded rock fill, with friction coefficient 0.40, command is:

```plaintext
add element # 1 type StressBasedSoftContact_NonLinHardShear
  with nodes ( 1, 2)
  initial_axial_stiffness = 2000 * MPa
  stiffening_rate = 100
  max_axial_stiffness = 10 * GPa
  initial_shear_stiffness = 8 * MPa
  axial_viscous_damping = 100 * Pa * s
  shear_viscous_damping = 100 * Pa * s
  residual_friction_coefficient = 0.40
  shear_zone_thickness = 5e-3*m
  contact_plane_vector = (0, 0, 1);
```

## 512.9.3 Formed Concrete Against Sand, Gravel and Rockfill

Recommended material parameters for frictional contact/interface of formed concrete against sand and gravel are given in Tables 512.3. Frictional properties given below are recommended by NAVFAC (1986).

Table 512.3: Friction coefficients for contact/interface of dissimilar materials, formed concrete against soil.

<table>
<thead>
<tr>
<th>Formed concrete against soil</th>
<th>Friction coefficient ((\tan \phi))</th>
<th>Friction angle ((\phi))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean gravel, sand-gravel mix, well graded rock fill</td>
<td>0.40 - 0.50</td>
<td>22° - 27°</td>
</tr>
<tr>
<td>Clean sand, silty sand-gravel mix, single size rock fill</td>
<td>0.30 - 0.40</td>
<td>17° - 22°</td>
</tr>
<tr>
<td>Silty sand, gravel or sand mixed with silt and clay</td>
<td>0.30</td>
<td>17°</td>
</tr>
<tr>
<td>Fine sandy silt, nonplastic silt</td>
<td>0.25</td>
<td>14°</td>
</tr>
</tbody>
</table>

Example command for contact/interface element for formed concrete against clean gravel, sand-gravel mix, well graded rock fill (friction coefficient 0.40-0.50) is given below:

```plaintext
add element # 1 type StressBasedSoftContact_NonLinHardShear
  with nodes ( 1, 2)
  initial_axial_stiffness = 30 * MPa
  stiffening_rate = 100
  max_axial_stiffness = 150 * MPa
  initial_shear_stiffness = 120 * kPa
  axial_viscous_damping = 100 * Pa * s
```
shear_viscous_damping = 100 * Pa * s
residual_friction_coefficient = 0.40
shear_zone_thickness = 5e-3*m
contact_plane_vector = (0, 0, 1);

512.10 Earthquake Motion Modeling

512.10.1 One Component (1C) Seismic Motions Defined at Surface or at Depth

One can add DRM loading directly, where input.hdf5 specifies the DRM motions to all DRM nodes.

```plaintext
add load # 1 type DRM
   hdf5_file = "input.hdf5"
   scale_factor = 1.0 ;
```

Since the direct specification of DRM motions to all DRM nodes is complicated, alternatively, user is able to specify DRM motion using a surface motion. Internally, wave deconvolution is conducted to specify the DRM motions to all DRM nodes.

```plaintext
add wave field # 1 with
   acceleration_filename = "acceleration.txt"
   unit_of_acceleration = 1 * m/s^2
   displacement_filename = "displacement.txt"
   unit_of_displacement = 1 * m
   add_compensation_time = 0.0 * s
   motion_depth = 0 * m
   monitoring_location = within_soil_layer
   soil_profile_filename = "soil_profile.txt"
   unit_of_Vs = 1 * m/s
   unit_of_rho = 1 * kg/m^3
   unit_of_damping = absolute
   unit_of_thickness = 1*m

add load # 1 type DRM from wave field # 1 in direction ux
   soil_surface at z = 0.0*m
   hdf5_file = "input.hdf5" ;
```

where input.hdf5 specifies the HDF5 file which contain the information about the DRM elements and DRM nodes.
512.10.2 3 × 1C Seismic Motions Defined at Surface or at Depth

One Example of add DRM load from wave fields:

```plaintext
add load # 1 type DRM from wave field
# 1 in direction ux
# 2 in direction uy
# 3 in direction uz
soil_surface at z = 0.0*m
hdf5_file = "input.hdf5";
```

512.10.3 Seismic Motions Imposed at Model Base

```plaintext
add load # 1 type imposed motion to node # 1 dof ux
time_step = 0.01*s
displacement_scale_unit = 1*m
displacement_file = "displacement.txt"
velocity_scale_unit = 1*m/s
velocity_file = "velocity.txt"
acceleration_scale_unit = 1*m/s^2
acceleration_file = "acceleration.txt";
```

512.10.4 Eigen Analysis

For structural model alone.

```plaintext
simulate using eigen algorithm
   number_of_modes = 3;
```

512.11 Core Modeling and Simulation Commands: Simulation Parameters

Developed model, using core functionality, as described above, numerically simulated using core functionality simulation controls.

Finite element system of equations can be solved in sequential processing mode, for smaller models, on sequential, single CPU computers (laptops, desktops, single CPU Amazon Web Services computers, etc.):

```plaintext
define solver sequential umfpack;
```
For larger models, parallel processing mode, on parallel computers (multi CPU laptops, multi CPU desktops, clusters of PCs, Amazon Web Services parallel computers, Supercomputers, etc.):

Command Example for a direct solver:

```plaintext
define solver parallel petsc "-pc_type lu -pc_factor_mat_solver_package mumps" ;
```

For selfweight phase of loading, static solution algorithm is used:

```plaintext
simulate 100 steps using static algorithm;
```

For static loading, for example self weight as described above, load application and the simulation process is controlled through load control:

```plaintext
define load factor increment 0.01;
```

For dynamic loads, simulation process is controlled using Newmark time integration method:

```plaintext
define dynamic integrator Newmark with gamma = 0.6000 beta = 0.3025;
```

The dynamic simulation process is performed in a number of steps:

```plaintext
simulate 2000 steps using transient algorithm time_step = 0.01*s;
```

For proper integration of constitutive equations on the integration point (Gauss point) level, within each finite element, constitutive algorithm needs to be defined:

```plaintext
define NDMaterial constitutive integration algorithm Forward_Euler;
```

For the finite element level, analysis of nonlinear systems require definition of nonlinear iteration algorithm:

```plaintext
define algorithm With_no_convergence_check;
```
Part 600

References
Bibliography


Z. Bai. Class notes on large scale scientific computing. ECS 231, Department of Computer Science, UC Davis, 2007.


W. F. Chen and D. J. Han. Plasticity for Structural Engineers. Springer Verlag, 1988a.

W. F. Chen and D. J. Han. Plasticity for Structural Engineers. Springer-Verlag, 1988b.


M. Čubrinovski. Private communications. ..., 2007 –.


B. Jeremić. Dynamic analysis of axisymmetric solids subjected to non-symmetric loading by the finite element method. Diploma thesis., Faculty of Civil Engineering, Belgrade University, July 1989. in Serbian. (Boris Jeremić Динамичка Анализа Ротационо Симетричних Тела оптерећених Несиметричним Оптерећењима Методом Коначних Елемената, Дипломски Рад Грађевински Факултет Универзитета у Београду.


G. Karypis and V. Kumar. METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 4.0. Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, September 1998c.

G. Karypis and V. Kumar. METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 4.0. Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, September 1998d.

G. Karypis, K. Schloegel, and V. Kumar. PARMETIS Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 3.1. Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, August 2003.


A. Niemunis. Private correspondence, 2015 –.


Various Authors. The C++ report: Columns on C++, 1991-.


Part 700

Appendix
Appendix 701

Useful Formulae

(1985-1989-1993-2021-
701.1 Chapter Summary and Highlights

701.2 Stress and Strain

This section reviews small deformation stress and strain measures used in this report.

701.2.1 Stress

In this work, the tensile stress is assumed positive, and in general we follow classical strength of materials (mechanics of materials) conventions for stress and strain. The stress tensor $\sigma_{ij}$ is defined as

$$\sigma_{ij} = \lim_{A_i \to 0} \frac{F_j}{A_i}$$  \hspace{1cm} (701.1)

where $F_j$ is a traction (force) in the $j$ direction and $A_i$ is an infinitesimal surface area with normal in $i$ direction. Cauchy stress tensor has a total of nine components, six of which are independent (symmetry $\sigma_{ij} = \sigma_{ji}$):

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} = \begin{pmatrix} \sigma_x & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z \end{pmatrix}$$  \hspace{1cm} (701.2)

In small deformation theory, this stress is symmetric, that is, $\sigma_{xy} = \sigma_{yx}, \sigma_{yz} = \sigma_{zy},$ and $\sigma_{zx} = \sigma_{xz}$. There are only six independent components and sometimes the stress can be expressed in the vector form

$$\sigma = \{\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{yz}, \sigma_{zx}\}$$  \hspace{1cm} (701.3)

The principle stresses $\sigma_1, \sigma_2, \text{ and } \sigma_3$ ($\sigma_1 \geq \sigma_2 \geq \sigma_3$) are the eigenvalues of the symmetric tensor $\sigma_{ij}$ in Equation 701.2 and can be obtained by solving the equation

$$\begin{vmatrix} \sigma_{xx} - \sigma & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} - \sigma & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} - \sigma \end{vmatrix} = 0$$  \hspace{1cm} (701.4)

or in alternative form

$$\sigma^3 - I_1\sigma^2 - I_2\sigma - I_3 = 0$$  \hspace{1cm} (701.5)

The three first-type stress invariants are then

$$I_1 = \sigma_{ii} = \sigma_{xx} + \sigma_{yy} + \sigma_{zz} = \sigma_1 + \sigma_2 + \sigma_3$$  \hspace{1cm} (701.6)
\[ I_2 = \frac{1}{2} \sigma_{ij} \sigma_{ji} \]
\[ = - (\sigma_{xx} \sigma_{yy} + \sigma_{yy} \sigma_{zz} + \sigma_{zz} \sigma_{xx}) + (\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2) \]
\[ = - (\sigma_1 \sigma_2 + \sigma_2 \sigma_3 + \sigma_3 \sigma_1) \quad (701.7) \]
\[ I_3 = \frac{1}{3} \sigma_{ij} \sigma_{jk} \sigma_{ki} = \det (\sigma_{ij}) \]
\[ = \sigma_{xx} \sigma_{yy} \sigma_{zz} + 2 \sigma_{xy} \sigma_{yz} \sigma_{zx} - (\sigma_{xx} \sigma_{y}^2 + \sigma_{yz} \sigma_{z}^2 + \sigma_{zx} \sigma_{x}^2) \]
\[ = \sigma_1 \sigma_2 \sigma_3 \quad (701.8) \]

The stress \( \sigma_{ij} \) can be decomposed into the hydrostatic stress \( \sigma_m \delta_{ij} \) and deviatoric stress \( s_{ij} \), with the definitions
\[
\sigma_m = \frac{1}{3} I_1, \quad s_{ij} = \sigma_{ij} - \frac{1}{3} \sigma_{kk} \delta_{ij} \quad (701.9)
\]
where \( \delta_{ij} \) is the Kronecker operator such that \( \delta_{ij} = 1 \) for \( i = j \) and \( \delta_{ij} = 0 \) for \( i \neq j \).

Since both hydrostatic and deviatoric stresses are stress tensors, they have their own coordinate-independent stress invariants respectively. The three invariants of the hydrostatic stress are
\[ I_1 = 3 \sigma_m = I_1, \quad I_2 = - 3 \sigma_m^2 = - \frac{1}{3} I_1^2, \quad I_3 = \sigma_m^3 = \frac{1}{27} I_1^3 \quad (701.10) \]

Since \( I_1, I_2 \) and \( I_3 \) are all simple functions of \( I_1 \), the hydrostatic stress state can therefore be represented by only one variable \( I_1 \).

The three eigenvalues of the deviatoric stresses \( s_{ij} \) are called principal deviatoric stresses, with the order \( s_1 \geq s_2 \geq s_3 \). The three invariants of the deviatoric stress are
\[ J_1 = s_{ii} = 0 \quad (701.11) \]
\[ J_2 = \frac{1}{2} s_{ij} s_{ji} \]
\[ = \frac{1}{3} I_1^2 + I_2 \]
\[ = \frac{1}{6} [(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2] \]
\[ = -(s_{xx} s_{yy} + s_{yy} s_{zz} + s_{zz} s_{xx}) + (s_{xy}^2 + s_{yz}^2 + s_{zx}^2) \]
\[ = \frac{1}{2} (s_1^2 + s_2^2 + s_3^2) = -(s_1 s_2 + s_2 s_3 + s_3 s_1) \quad (701.12) \]
\[ J_3 = \frac{1}{3} s_{ij} s_{jk} s_{ki} = \det (s_{ij}) \]
\[ = I_3 + \frac{1}{3} I_1 I_2 + \frac{2}{27} I_1^3 = I_3 - \frac{1}{3} I_1 J_2 - \frac{1}{27} I_1^3 \]
\[ = \frac{1}{27} (2 \sigma_1 - \sigma_2 - \sigma_3)(2 \sigma_2 - \sigma_3 - \sigma_1)(2 \sigma_3 - \sigma_1 - \sigma_2) \]
\[ = s_{xx} s_{yy} s_{zz} + 2 s_{xy} s_{yz} s_{zx} - (s_{xx} s_{y}^2 + s_{yy} s_{z}^2 + s_{zz} s_{x}^2) \]
\[ = s_1 s_2 s_3 \quad (701.13) \]
The deviatoric stress state can therefore be represented by only two variables \( J_2 \) and \( J_3 \).

Combining hydrostatic and deviatoric stress, we can conclude that the stress state can be represented by three variables \( I_1, J_2 \) and \( J_3 \). Using the three invariants \((I_1, J_2, J_3)\) or its equivalents instead of the nine components of \( \sigma_{ij} \) is widely used in geomechanics.

The stress state may also be described in three dimensional space \((p, q, \theta_\sigma)\), defined as

\[
\begin{align*}
p &= -\frac{1}{3}I_1 \\
q &= \sqrt{3}J_2 \\
\theta_\sigma &= \frac{1}{3} \arccos \left( \frac{3\sqrt{3}J_3}{2\sqrt{J_3^2}} \right)
\end{align*}
\]

where \( \theta_{\sigma ij} \) is the stress Lode’s angle \((0 \leq \theta_{\sigma ij} \leq \pi/3)\). A stress state with \( \theta_\sigma = 0 \) corresponds to the meridian of conventional triaxial compression (CTC), while \( \theta_\sigma = \pi/3 \) to the meridian of conventional triaxial extension (CTE). The relationship between \((\sigma_1, \sigma_2, \sigma_3)\) and \((p, q, \theta_\sigma)\) is

\[
\begin{pmatrix}
\sigma_1 \\
\sigma_2 \\
\sigma_3
\end{pmatrix} = -p + \frac{2}{3} q \begin{pmatrix}
cos \theta_\sigma \\
\cos (\theta_\sigma - \frac{2}{3}\pi) \\
\cos (\theta_\sigma + \frac{2}{3}\pi)
\end{pmatrix}
\]

The line of the principal stress space diagonal is called hydrostatic axis. Any plane perpendicular to the hydrostatic axis is an deviatoric plane, or \( \pi \) plane. The Haigh-Westergaard three dimensional stress coordinate system \((\xi, \rho, \theta_\sigma)\) Chen and Han (1988a), is defined as

\[
\begin{align*}
\xi &= \frac{I_1}{\sqrt{3}} = -\sqrt{3}p \\
\rho &= \sqrt{2J_2} = \sqrt{\frac{2}{3}}q
\end{align*}
\]

The Haigh-Westergaard invariants have physical meanings. \( \xi \) is the distance of the deviatoric plane to the origin of the Haigh-Westergaard coordinates, and \( \rho \) is the distance of a stress point to the hydrostatic line and represents the magnitude of the deviatoric stress. The projections of the axes \( \sigma_1, \sigma_2 \) and \( \sigma_3 \) on the deviatoric plane are assumed \( \sigma'_1, \sigma'_2 \) and \( \sigma'_3 \) respectively. \((\rho, \theta_\sigma)\) is the polar coordinate system in the deviatoric plane with the \( \sigma'_1 \) the polar axis and \( \theta_\sigma \) the polar angle. The relationship between \((\sigma_1, \sigma_2, \sigma_3)\) and \((\xi, \rho, \theta_\sigma)\) is

\[
\begin{pmatrix}
\sigma_1 \\
\sigma_2 \\
\sigma_3
\end{pmatrix} = \frac{1}{\sqrt{3}}\xi + \frac{2}{3}\rho \begin{pmatrix}
cos \theta_\sigma \\
\cos (\theta_\sigma - \frac{2}{3}\pi) \\
\cos (\theta_\sigma + \frac{2}{3}\pi)
\end{pmatrix}
\]
701.2.2 Strain

Point \( P(x_i) \) and nearby point \( Q(x_i + \text{d}x_i) \) displace due to applied loading to new positions \( P(x_i + U_i) \) and \( Q(u_i + (\partial u_i/\partial x_j)\text{d}x_j) \). We can define a displacement gradient tensor \( u_{i,j} \) as

\[
  u_{i,j} = \frac{\partial u_i}{\partial x_j}
\]

(701.21)

Matrix form of the displacement gradient can decomposed into the symmetric and antisymmetric parts

\[
\begin{pmatrix}
  u_{1,1} & u_{1,2} & u_{1,3} \\
  u_{2,1} & u_{2,2} & u_{2,3} \\
  u_{3,1} & u_{3,2} & u_{3,3}
\end{pmatrix} = \begin{pmatrix}
  u_{1,1} & \frac{1}{2}(u_{1,2} + u_{2,1}) & \frac{1}{2}(u_{1,3} + u_{3,1}) \\
  \frac{1}{2}(u_{2,1} + u_{1,2}) & u_{2,2} & \frac{1}{2}(u_{2,3} + u_{3,2}) \\
  \frac{1}{2}(u_{3,1} + u_{1,3}) & \frac{1}{2}(u_{3,2} + u_{2,3}) & u_{3,3}
\end{pmatrix}
\]

\[
+ \begin{pmatrix}
  0 & \frac{1}{2}(u_{1,2} - u_{2,1}) & \frac{1}{2}(u_{1,3} - u_{3,1}) \\
  \frac{1}{2}(u_{2,1} - u_{1,2}) & 0 & \frac{1}{2}(u_{2,3} - u_{3,2}) \\
  \frac{1}{2}(u_{3,1} - u_{1,3}) & \frac{1}{2}(u_{3,2} - u_{2,3}) & 0
\end{pmatrix}
\]

(701.22)

or

\[
u_{i,j} = \epsilon_{ij} + w_{ij}
\]

(701.23)

where

\[
\epsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i})
\]

(701.24)

\[
w_{ij} = \frac{1}{2}(u_{i,j} - u_{j,i})
\]

(701.25)

The symmetric part of the deformation gradient tensor, \( \epsilon_{ij} \), is the small deformation strain tensor \(^1\), while the antisymmetric part of the deformation gradient tensor, \( w_{ij} \), is the rotation motion tensor. The matrix form of the strain \( \epsilon_{ij} \) is

\[
\epsilon = \begin{pmatrix}
  \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\
  \epsilon_{xy} & \epsilon_{yy} & \epsilon_{yz} \\
  \epsilon_{xz} & \epsilon_{yz} & \epsilon_{zz}
\end{pmatrix} = \begin{pmatrix}
  \epsilon_x & \frac{1}{2}\gamma_{xy} & \frac{1}{2}\gamma_{zx} \\
  \frac{1}{2}\gamma_{xy} & \epsilon_y & \frac{1}{2}\gamma_{yz} \\
  \frac{1}{2}\gamma_{zx} & \frac{1}{2}\gamma_{yz} & \epsilon_z
\end{pmatrix}
\]

(701.26)

The engineering strain is usually expressed in the vector form

\[
\epsilon = \{\epsilon_x, \epsilon_y, \epsilon_z, \gamma_{xy}, \gamma_{yz}, \gamma_{zx}\}^T
\]

(701.27)

Note that the engineering shear strain \( \gamma_{ij} \) is the double of the corresponding strain component \( \epsilon_{ij} \).

\(^1\)Here the second and higher order derivative terms are neglected due to the small deformation assumption.
Similar to the stress tensor, the strain tensor also has three principle strains \( \epsilon_i (\epsilon_1 \geq \epsilon_2 \geq \epsilon_3) \), and three strain invariants \( I'_1, I'_2, \text{ and } I'_3 \), defined as

\[
I'_1 = \epsilon_{ii} = \epsilon_v = \epsilon_{xx} + \epsilon_{yy} + \epsilon_{zz} = \epsilon_1 + \epsilon_2 + \epsilon_3 \tag{701.28}
\]

\[
I'_2 = \frac{1}{2} \epsilon_{ij} \epsilon_{ji} = - (\epsilon_{xx} \epsilon_{yy} + \epsilon_{yy} \epsilon_{zz} + \epsilon_{zz} \epsilon_{xx}) + (\epsilon_{xy}^2 + \epsilon_{yz}^2 + \epsilon_{zx}^2)
\]

\[
= -(\epsilon_1 \epsilon_2 + \epsilon_2 \epsilon_3 + \epsilon_3 \epsilon_1) \tag{701.29}
\]

\[
I'_3 = \frac{1}{3} \epsilon_{ij} \epsilon_{jk} \epsilon_{ki} = \det (\epsilon_{ij}) = \epsilon_{xx} \epsilon_{yy} \epsilon_{zz} + 2 \epsilon_{xy} \epsilon_{yz} \epsilon_{zx} - (\epsilon_{xx} \epsilon_{yz}^2 + \epsilon_{yy} \epsilon_{zx}^2 + \epsilon_{zz} \epsilon_{xy}^2)
\]

\[
= \epsilon_1 \epsilon_2 \epsilon_3 \tag{701.30}
\]

The first strain invariant is also called the volumetric strain \( \epsilon_v \).

The strain \( \epsilon_{ij} \) can be decomposed into the hydrostatic strain \( \epsilon_m \delta_{ij} \) and deviatoric strain \( e_{ij} \) through

\[
\epsilon_{ij} = \epsilon_m \delta_{ij} + e_{ij}
\]

where:

\[
\epsilon_m = \frac{1}{3} I'_1, \quad e_{ij} = \epsilon_{ij} - \frac{1}{3} \epsilon_{kk} \delta_{ij} \tag{701.31}
\]

Since both hydrostatic and deviatoric strains are strain tensors, they have their own strain invariants respectively. The three invariants of the hydrostatic strain are

\[
I'_1 = 3 \epsilon_m = I'_1, \quad I'_2 = - 3 \epsilon_m^2 = - \frac{1}{3} (I'_1)^2, \quad I'_3 = \epsilon_m^3 = \frac{1}{27} (I'_1)^3 \tag{701.32}
\]

The hydrostatic strain state can therefore be represented by only one variable \( I'_1 \).

The three eigenvalues of the deviatoric strains \( e_{ij} \) are called principal deviatoric strains, with the order \( e_1 \geq e_2 \geq e_3 \). The three invariants of the deviatoric strain are

\[
J'_1 = e_{ii} = 0 \tag{701.33}
\]

\[
J'_2 = \frac{1}{2} e_{ij} e_{ji} = \frac{1}{3} (I'_1)^2 + I'_2
\]

\[
= \frac{1}{6} [(\epsilon_1 - \epsilon_2)^2 + (\epsilon_2 - \epsilon_3)^2 + (\epsilon_3 - \epsilon_1)^2]
\]

\[
= -(\epsilon_{xx} \epsilon_{yy} + \epsilon_{yy} \epsilon_{zz} + \epsilon_{zz} \epsilon_{xx}) + (\epsilon_{xy}^2 + \epsilon_{yz}^2 + \epsilon_{zx}^2)
\]

\[
= \frac{1}{2} (e_1^2 + e_2^2 + e_3^2) = -(\epsilon_1 \epsilon_2 + \epsilon_2 \epsilon_3 + \epsilon_3 \epsilon_1) \tag{701.34}
\]
\[ J_3' = \frac{1}{3} e_{ij} e_{jk} e_{ki} = \det(e_{ij}) \]

\[ = I_3' + \frac{1}{3} I_1' I_2' + \frac{2}{27} (I_1')^3 = I_3 - \frac{1}{3} I_1' J_2' - \frac{1}{27} (I_1')^3 \]

\[ = \frac{1}{27} (2\epsilon_1 - \epsilon_2 - \epsilon_3)(2\epsilon_2 - \epsilon_3 - \epsilon_1)(2\epsilon_3 - \epsilon_1 - \epsilon_2) \]

\[ = e_{xx} e_{yy} e_{zz} + 2 e_{xy} e_{yz} e_{zx} - (e_{xx} e_{yz}^2 + e_{yy} e_{zx}^2 + e_{zz} e_{xy}^2) \]

\[ = e_1 e_2 e_3 \]  

(701.35)

The deviatoric strain state can therefore be represented by only two variables \( J_2' \) and \( J_3' \).

Combining the hydrostatic and deviatoric strain, we can conclude that the strain state can be represented by three variables \( I_1' \), \( J_2' \) and \( J_3' \).

Strain state may also be represented with another three invariant \((\epsilon_p, \epsilon_q, \theta_\epsilon)\), defined as

\[ \epsilon_p = -I_1' = -\epsilon_v \]  

(701.36)

\[ \epsilon_q = 2\sqrt{\frac{J_2'}{3}} \]  

(701.37)

\[ \theta_\epsilon = \frac{1}{3} \arccos \left( \frac{3\sqrt{3}/2}{\sqrt{(J_3')^3}} \right) \]  

(701.38)

where \( \theta_\epsilon \) is the strain Lode’s angle and \( 0 \leq \theta_\epsilon \leq \pi/3 \). The relationship between \((\epsilon_1, \epsilon_2, \epsilon_3)\) and \((\epsilon_p, \epsilon_q, \theta_\epsilon)\) is

\[
\begin{pmatrix}
\epsilon_1 \\
\epsilon_2 \\
\epsilon_3
\end{pmatrix} = -\frac{1}{3} \epsilon_p + \sqrt{\frac{3}{2}} \epsilon_q \begin{pmatrix}
\cos \theta_\epsilon \\
\cos (\theta_\epsilon - \frac{2}{3} \pi) \\
\cos (\theta_\epsilon + \frac{2}{3} \pi)
\end{pmatrix}
\]  

(701.39)

701.3 Derivatives of Stress Invariants

In this part of the Appendix, we shall derive some useful formulae, that are rarely found in texts treating elasto–plastic problems in mechanics of solid continua.

First derivative of \( I_1 \) with respect to stress tensor \( \sigma_{ij} \):

\[
\frac{\partial I_1}{\partial \sigma_{ij}} = \frac{\partial \sigma_{kk}}{\partial \sigma_{ij}} = \delta_{ij}
\]

First derivative of \( J_{2D} \) with respect to stress tensor \( \sigma_{ij} \):

\[ ^2 \text{if found at all.} \]
First derivative of $J_{3D}$ with respect to stress tensor $\sigma_{pq}$:

$$\frac{\partial J_{3D}}{\partial \sigma_{pq}} = \frac{\partial}{\partial \sigma_{pq}} \left( \frac{1}{3} \sigma_{pq} \delta_{kk} \right) = \frac{1}{3} \delta_{pq} \sigma_{kk} = \frac{1}{3} \delta_{pq} \sigma_{kk}$$

First derivative of $s_{pq}$ with respect to stress tensor $\sigma_{mn}$, or second derivative of $J_{2D}$ with respect to stress tensors $\sigma_{pq}$ and $\sigma_{mn}$:

$$\frac{\partial s_{pq}}{\partial \sigma_{mn}} = \frac{\partial}{\partial \sigma_{mn}} \left( \sigma_{pq} - \frac{1}{3} \delta_{pq} \delta_{kk} \right) = \frac{\partial}{\partial \sigma_{mn}} \left( \delta_{mp} \delta_{nq} - \frac{1}{3} \delta_{pq} \delta_{mn} \right) = p_{pqmn}$$

Second derivative of $J_{3D}$ with respect to stress tensors $\sigma_{pq}$ and $\sigma_{mn}$:

1 because $\delta_{mn} \delta_{ij} s_{mn} \equiv 0$

2 since $\frac{1}{3} \delta_{ij} \delta_{kp} \delta_{qk} s_{jk} s_{ki} = \frac{1}{3} \delta_{kp} \delta_{qk} s_{ik} s_{ki} = \frac{1}{3} \delta_{kp} \delta_{qk} s_{ik} s_{ki} = \frac{2}{3} \delta_{pq} J_{2D}$ see also Chen and Han (1988a) page 222
\[
\frac{\partial t_{pq}}{\partial \sigma_{mn}} = \frac{\partial}{\partial \sigma_{mn}} \left( s_{qk}s_{kp} - \frac{2}{3} \delta_{pq} J_{2D} \right) = \frac{\partial}{\partial \sigma_{mn}} \left( s_{qk}s_{kp} \right) - \frac{\partial}{\partial \sigma_{mn}} \left( \frac{2}{3} \delta_{pq} J_{2D} \right) = \\
= \frac{\partial}{\partial \sigma_{mn}} \left( s_{qk}s_{kp} \right) - \frac{2}{3} \delta_{pq} \frac{\partial J_{2D}}{\partial \sigma_{mn}} = \frac{\partial s_{qk}}{\partial \sigma_{mn}} s_{kp} + s_{qk} \frac{\partial s_{kp}}{\partial \sigma_{mn}} - \frac{2}{3} \delta_{pq} \frac{\partial s_{mn}}{\partial \sigma_{mn}} = \\
= \left( \delta_{qm} \delta_{nk} - \frac{1}{3} \delta_{qk} \delta_{nm} \right) s_{kp} + s_{qk} \left( \delta_{km} \delta_{np} - \frac{1}{3} \delta_{kp} \delta_{nm} \right) - \frac{2}{3} \delta_{pq} \delta_{mn} = \\
= \left( \delta_{qm} s_{np} - \frac{1}{3} s_{qp} \delta_{nm} \right) + \left( s_{qm} \delta_{np} - \frac{1}{3} s_{qp} \delta_{nm} \right) - \frac{2}{3} \delta_{pq} \delta_{mn} = \\
= s_{np} \delta_{qm} + s_{qm} \delta_{np} - \frac{2}{3} s_{qp} \delta_{nm} - \frac{2}{3} \delta_{pq} \delta_{mn} = w_{pqmn}
\]

Multiplying stiffness tensor \( E_{ijkl} \) with compliance tensor \( D_{klpq} \):
\[
\frac{E}{2(1 + \nu)} \left( \frac{1}{1 - 2\nu} \frac{2\nu}{1 + \nu} \delta_{ij} \delta_{kl} + \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} \right) \left( \frac{1}{1 + \nu} \frac{2\nu}{1 - 2\nu} \delta_{kl} \delta_{pq} + \delta_{kp} \delta_{lq} + \delta_{kq} \delta_{lp} \right) = \]
\[
\frac{1}{4} \left( \delta_{ik} \delta_{jl} \delta_{kp} \delta_{lq} + \delta_{il} \delta_{jk} \delta_{kp} \delta_{lq} + \delta_{ik} \delta_{jl} \delta_{kq} \delta_{lp} + \delta_{il} \delta_{jk} \delta_{kq} \delta_{lp} \right) + \]
\[
\frac{\nu}{2(1 - 2\nu)} \left( \delta_{ij} \delta_{kl} \delta_{pq} + \delta_{ij} \delta_{kl} \delta_{pq} \right) - \frac{\nu}{2(1 + \nu)} \left( \delta_{kl} \delta_{pq} + \delta_{kl} \delta_{pq} \right) - \]
\[
\frac{\nu^2}{(1 - 2\nu)(1 + \nu)} \delta_{ij} \delta_{kl} \delta_{pq} = \]
\[
\frac{1}{2} \left( \delta_{ip} \delta_{jq} + \delta_{iq} \delta_{jp} \right) + \]
\[
\frac{\nu}{2(1 - 2\nu)} \left( \delta_{ij} \delta_{kq} + \delta_{ij} \delta_{kp} \right) - \frac{\nu}{2(1 + \nu)} \left( \delta_{ij} \delta_{pq} + \delta_{ij} \delta_{pq} \right) - \]
\[
\frac{3\nu^2}{(1 - 2\nu)(1 + \nu)} \delta_{ij} \delta_{pq} = \]
\[
\frac{1}{2} \left( \delta_{ip} \delta_{jq} + \delta_{iq} \delta_{jp} \right) + \]
\[
\frac{\nu}{2(1 - 2\nu)} \delta_{ij} \delta_{pq} - \frac{\nu}{2(1 + \nu)} \delta_{ij} \delta_{pq} - \]
\[
\frac{3\nu^2}{(1 - 2\nu)(1 + \nu)} \delta_{ij} \delta_{pq} = \]
\[
\frac{1}{2} \left( \delta_{ip} \delta_{jq} + \delta_{iq} \delta_{jp} \right) + \]
\[
\frac{\nu}{1 - 2\nu} \delta_{ij} \delta_{pq} - \frac{\nu}{2(1 + \nu)} \delta_{ij} \delta_{pq} - \]
\[
\frac{3\nu^2}{(1 - 2\nu)(1 + \nu)} \delta_{ij} \delta_{pq} = \]
\[
\frac{1}{2} \left( \delta_{ip} \delta_{jq} + \delta_{iq} \delta_{jp} \right) + \]
\[
\frac{\nu(1 + \nu) - \nu(1 - 2\nu) - 3\nu^2}{(1 - 2\nu)(1 + \nu)} \delta_{ij} \delta_{pq} = \]
\[
\frac{1}{2} \left( \delta_{ij} \delta_{pq} + \delta_{ij} \delta_{pq} \right) = I_{ijpq}^{\text{sym}} \]
Appendix 702

The nDarray Programming Tool

702.1 Chapter Summary and Highlights

Material in this chapter is based on the following publications Jeremić (1993); Jeremić and Sture (1998).

This section describes a programming tool, \textit{nDarray}, which is designed using an Object Oriented Paradigm (OOP) and implemented in the C++ programming language. Finite element equations, represented in terms of multidimensional tensors are easily manipulated and programmed. The usual matrix form of the finite element equations are traditionally coded in FORTRAN, which makes it difficult to build and maintain complex program systems. Multidimensional data systems and their implementation details are seldom transparent and thus not easily dealt with and usually avoided. On the other hand, OOP together with efficient programming in C++ allows building new concrete data types, namely tensors of any order, thus hiding the lower level implementation details. These concrete data types prove to be quite useful in implementing complicated tensorial formulae associated with the numerical solution of various elastic and elastoplastic problems in solid mechanics. They permit implementing complex nonlinear continuum mechanics theories in an orderly manner. Ease of use and the immediacy of the \textit{nDarray} programming tool in constitutive driver programming and in building finite element classes will be shown.

702.2 Introduction

In implementing complex programming systems for finite element computations, the analyst is usually faced with the challenge of transforming complicated tensorial formulae to a matrix form. Considerable amount of time in solving problems by the finite element method is often devoted to the actual implementation process. If one decides to use FORTRAN, a number of finite element and numerical libraries are readily available. Although quick results can be produced in solving simpler problems, when implementing complex small deformation elastoplastic or large deformation elastic and elastoplastic algorithms, C++ provides clear benefits.

Some of the improvements C++ provides over C and FORTRAN are classes for encapsulating abstractions, the possibility of building user-defined concrete data types and operator overloading for expressing complex formulae in a natural way. In the following we shall show that the \textit{nDarray} tool will allow analysts to be a step closer to the problem space and a step further away from the underlying machine.

As most analysts know, the intention (Stroustrup, 1994) behind C++ was not to replace C. Instead, C was extended with far more freedom given to the program designer and implementor. In C and FORTRAN, large applications become collections of programs and functions, order and the structure are left to the programmer. The C++ programming language embodies the OOP, which can be used to
simplify and organize complex programs. One can build a hierarchy of derived classes and nest classes inside other classes. A concern in C and FORTRAN programming languages is handling data type conflicts and data which are being operated on or passed. The C++ programming language extends the definition of type to include abstract data types. With abstract data types, data can be encapsulated with the methods that operate on it. The C++ programming language offers structure and mechanisms to handle larger, more complex programming systems. Object Oriented technology, with function and operator overloading, inheritance and other features, provides means of attacking a problem in a natural way. Once basic classes are implemented, one can concentrate on the physics of a problem. By building further abstract data types one can describe the physics of a problem rather than spend time on the lower level programming issues. One should keep in mind the adage, credited to the original designer and implementor of C++ programming language, Bjarne Stroustrup: “C makes it easy to shoot yourself in the foot, C++ makes it harder, but when you do, it blows away your whole leg”.

Rather than attempting here to give a summary of Object Oriented technology we will suggest useful references for readers who wish to explore the subject in greater depth (Booch, 1994). The current language definition is given in the Working Paper for Draft Proposed International Standard for Information Systems–Programming Language C++ (ANS, 1995). Detailed description of language evolution and main design decisions are given by Stroustrup (1994). Useful sets of techniques, explanations and directions for designing and implementing robust C++ code are given in books (Coplien, 1992) (Eckel, 1989) and journal articles (Koenig, 1989 - 1993) (Various Authors, 1991-).

Increased interest in using Object Oriented techniques for finite element programming has resulted in a number (Donescu and Laursen, 1996) (Eyheramendy and Zimmermann, 1996) (Forde et al., 1990) (Miller, 1991) (Pidaparti and Hudli, 1993) (Scholz, 1992) (Zeglinski et al., 1994) of experimental developments and implementations. Programming techniques used in some of the papers are influenced by the FORTRAN programming style. Examples provided in some of the above mentioned papers are readable by C++ experts only. It appears that none of the authors have used Object Oriented techniques for complex elastoplasticity computations.

702.3 nDarray Programming Tool

702.3.1 Introduction to the nDarray Programming Tool

The nDarray programming tool is a set of classes written in the C++ programming language. The main purpose of the package is to facilitate algebraic manipulations with matrices, vectors and tensors that are often found in computer codes for solving engineering problems. The package is designed and implemented using the Object Oriented philosophy. Great care has been given to the problem of
cross-platform and cross-compiler portability. Currently, the \texttt{nDarray} set of classes has been tested and running under the following C++ compilers:

- Sun CC on SunOS and Solaris platforms,
- IBM \texttt{xlC} on AIX RISC/6000 platforms,
- Borland C++ and Microsoft C++ on DOS/Windows platforms,
- CodeWarrior C++ on Power Macintosh platform,
- GNU g++ on SunOS, SOLARIS, LINUX, AIX, HPUX and AMIGA platforms.

### 702.3.2 Abstraction Levels

\texttt{nDarray} tool has the following simple class hierarchy:

```
    nDarray_rep, nDarray
    matrix
    vector
    tensor
```

Indentation of class names implies the inheritance level. For example, class vector is derived from class matrix, which, in turn is derived from classes \texttt{nDarray} and \texttt{nDarray\_rep}. The idea is to subdivide classes into levels of abstraction, and hide the implementation from end users. This means that the end user can use the \texttt{nDarray} tool on various levels.

- At the highest level of abstraction, one can use tensor, matrix and vector objects without knowing anything about the implementation and the inner workings. They are all designed and implemented as concrete data types. In spite of the very powerful code that can be built using Object Oriented technology, it would be unwise to expect proficiency in Object Oriented techniques and the C++ programming language from end users. It was our aim to provide power programming with multidimensional data types to users with basic knowledge of C.

- At a lower abstraction level, users can address the task of the actual implementation of operators and functions for vector, matrix and tensor classes. A number of improvements can be made, especially in optimizing some of the operators.

- The lowest level of abstraction is associated with \texttt{nDarray} and \texttt{nDarray\_rep} classes. Arithmetic operators\footnote{Like addition and subtraction.} are implemented at this level.
Next, classes are described from the base and down the inheritance tree. Later we focus our attention on `nDarray` usage examples. Our goal is to provide a useful programming tool, rather than to teach OOP or to show C++ implementation. For readers interested in actual implementation details, source code, examples and makefiles are available at http://sokocalo.engr.ucdavis.edu/~jeremic.

702.3.2.1 nDarray_rep class

The nDarray_rep class is a data holder and represents an \( n \) dimensional array object. A simple memory manager, implemented with the reference counting idiom (Coplien, 1992) is used. The memory manager uses rather inefficient built-in C memory allocation functions. Performance can be improved if one designs and implements specially tailored allocation functions for fast heap manipulations. Another possible improvement is in using memory resources other than heap memory. Sophisticated memory management introduced by the reference counting is best explained by Coplien (1992). The nDarray_rep class is not intended for stand-alone use. It is closely associated with the nDarray class.

The data structure of nDarray_rep introduces a minimal amount of information about a multidimensional array object. The actual data are stored as a one-dimensional array of double numbers. Rank, total number of elements, and array of dimensions are all that is needed to represent a multidimensional object. The data structure is allocated dynamically from the heap, and memory is reclaimed by the system after the object has gone out of scope.

702.3.2.2 nDarray class

The nDarray class together with the nDarray_rep class represents the abstract base for derived multidimensional data types: matrices, vectors and tensors. Objects derived from the nDarray class are generated dynamically by constructor functions at the first appearance of an object and are destroyed at the end of the block in which the object is referenced. The reference counting idiom provides for the object's life continuation after the end of the block where it was defined. To extend an object's life, a standard C++ compiler would by default call constructor functions, thus making the entire process of returning large objects from functions quite inefficient. By using reference counting idiom, destructor and constructor functions manipulate reference counter which results in a simple copying of a pointer to nDarray_rep object. By using this technique, copying of large objects is made very efficient.

Objects can be created from an array of values, or from a single scalar value, as shown in Table 702.1. Some of the frequently used multidimensional arrays are predefined and can be constructed by sending the proper flag to the constructor function. For example by sending the “I” flag one creates Kronecker delta \( \delta_{ij} \) and by sending “e” flag, one creates a rank 3 Levi-Civita permutation tensor \( e_{ijk} \). Functions and operators common to multidimensional data types are defined in the nDarray
Table 702.1: nDarray constructor functions.

class, as described in Table 702.2. These common operators and functions are inherited by derived classes. Occasionally, some of the functions will be redefined, overloaded in derived classes. In tensor multiplications we need additional information about indices. For example

\[ C_{il} = (A_{ijk} + B_{ijk}) * D_{jkl} \]

\[ C = (A("ijk") + B("ijk")) * D("jkl"), \]

to be used for multiplication with \( D_{jkl} \). It is interesting to note (Koenig, 1989 - 1993) that operator \( += \) is defined as a member and \( + \) is defined as an inline function in terms of \( += \) operator.

### 702.3.2.3 Matrix and Vector Classes

The matrix class is derived from the nDarray class through the public construct. It inherits common operators and functions from the base nDarray class, but it also adds its own set of functions and operators. Table (702.3) summarizes some of the more important additional functions and operators for the matrix class. The vector class defines vector objects and is derived and inherits most operators and data members from the matrix class. Some functions, like copy constructor, are overloaded in order to handle specifics of a vector object.

### 702.3.2.4 Tensor Class

The main goal of the tensor class development was to provide the implementing analyst with the ability to write the following equation directly into a computer program:

\[ d\sigma_{mn} = -\frac{\partial}{\partial \lambda} T^{-1}_{ijmn} - d\lambda E_{ijkl} m_{kl} T^{-1}_{ijmn} \]

as:

\[ dsigma = -(r("ij") * Tinv("ijmn")) - dlambda*((E("ijkl") * dQods("kl")) * Tinv("ijmn")); \]
<table>
<thead>
<tr>
<th>operator or function</th>
<th>left value</th>
<th>right value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>nDarray</td>
<td>nDarray</td>
<td>nDarray assignment</td>
</tr>
<tr>
<td>+</td>
<td>nDarray</td>
<td>nDarray</td>
<td>nDarray addition</td>
</tr>
<tr>
<td>+=</td>
<td>nDarray</td>
<td>nDarray</td>
<td>nDarray addition</td>
</tr>
<tr>
<td>-</td>
<td>nDarray</td>
<td></td>
<td>unary minus</td>
</tr>
<tr>
<td>-=</td>
<td>nDarray</td>
<td>nDarray</td>
<td>nDarray subtraction</td>
</tr>
<tr>
<td>*</td>
<td>double</td>
<td>nDarray</td>
<td>scalar multiplication (from left)</td>
</tr>
<tr>
<td>*=</td>
<td>nDarray</td>
<td>double</td>
<td>scalar multiplication (from right)</td>
</tr>
<tr>
<td>==</td>
<td>nDarray</td>
<td>nDarray</td>
<td>nDarray comparison</td>
</tr>
<tr>
<td>val(...)</td>
<td>nDarray</td>
<td></td>
<td>reference to members of nDarray</td>
</tr>
<tr>
<td>cval(...)</td>
<td>nDarray</td>
<td></td>
<td>members of nDarray</td>
</tr>
<tr>
<td>trace()</td>
<td>nDarray</td>
<td></td>
<td>trace of square nDarray</td>
</tr>
<tr>
<td>eigenvalues()</td>
<td>nDarray</td>
<td></td>
<td>eigenvalues of rank 2 square nDarray</td>
</tr>
<tr>
<td>eigenvectors()</td>
<td>nDarray</td>
<td></td>
<td>eigenvectors of rank 2 square nDarray</td>
</tr>
<tr>
<td>General_norm()</td>
<td>nDarray</td>
<td></td>
<td>general p-th norm of nDarray</td>
</tr>
<tr>
<td>nDsqrt()</td>
<td>nDarray</td>
<td></td>
<td>square root of nDarray</td>
</tr>
<tr>
<td>print(...)</td>
<td>nDarray</td>
<td></td>
<td>generic print function</td>
</tr>
</tbody>
</table>

Table 702.2: Public functions and operators for nDarray class.

<table>
<thead>
<tr>
<th>operator or function</th>
<th>left value</th>
<th>right value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>matrix</td>
<td>matrix</td>
<td>matrix assignment</td>
</tr>
<tr>
<td>*</td>
<td>matrix</td>
<td>matrix</td>
<td>matrix multiplication</td>
</tr>
<tr>
<td>transpose()</td>
<td>matrix</td>
<td></td>
<td>matrix transposition</td>
</tr>
<tr>
<td>determinant()</td>
<td>matrix</td>
<td></td>
<td>determinant of a matrix</td>
</tr>
<tr>
<td>inverse()</td>
<td>matrix</td>
<td></td>
<td>matrix inversion</td>
</tr>
</tbody>
</table>

Table 702.3: Matrix class functions and operators (added on nDarray class definitions).
Instead of developing theory in terms of indicial notation, then converting everything to matrix notation and then implementing it, we were able to copy formulae directly from their indicial form to the C++ source code.

In addition to the definitions in the base nDArray class, the tensor class adds some specific functions and operators. Table 702.4 summarizes some of the main new functions and operators. The most significant addition is the tensor multiplication operator. With the help of a simple indicial parser, the multiplication operator contracts or expands indices and yields a resulting tensor of the correct rank. The resulting tensor receives proper indices, and can be used in further calculations on the same code statement.

### Table 702.4: Additional and overloaded functions and operators for tensor class.

<table>
<thead>
<tr>
<th>operator or function</th>
<th>left value</th>
<th>right value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>tensor</td>
<td>tensor</td>
<td>tensor addition</td>
</tr>
<tr>
<td>-</td>
<td>tensor</td>
<td>tensor</td>
<td>tensor subtraction</td>
</tr>
<tr>
<td>*</td>
<td>tensor</td>
<td>tensor</td>
<td>tensor multiplication</td>
</tr>
<tr>
<td>transpose0110()</td>
<td>tensor</td>
<td>A&lt;sub&gt;ijkl&lt;/sub&gt; → A&lt;sub&gt;ikjl&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>transpose0101()</td>
<td>tensor</td>
<td>A&lt;sub&gt;ijkl&lt;/sub&gt; → A&lt;sub&gt;ilkj&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>transpose0111()</td>
<td>tensor</td>
<td>A&lt;sub&gt;ijkl&lt;/sub&gt; → A&lt;sub&gt;iljk&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>transpose1100()</td>
<td>tensor</td>
<td>A&lt;sub&gt;ijkl&lt;/sub&gt; → A&lt;sub&gt;jikl&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>transpose0011()</td>
<td>tensor</td>
<td>A&lt;sub&gt;ijkl&lt;/sub&gt; → A&lt;sub&gt;ijlk&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>transpose1001()</td>
<td>tensor</td>
<td>A&lt;sub&gt;ijkl&lt;/sub&gt; → A&lt;sub&gt;lijk&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>transpose11()</td>
<td>tensor</td>
<td>a&lt;sub&gt;ij&lt;/sub&gt; → a&lt;sub&gt;ji&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>symmetrize11()</td>
<td>tensor</td>
<td>symmetrize second order tensor</td>
<td></td>
</tr>
<tr>
<td>determinant()</td>
<td>tensor</td>
<td>determinant of 2nd order tensor</td>
<td></td>
</tr>
<tr>
<td>inverse()</td>
<td>tensor</td>
<td>tensor inversion (2nd, 4th order)</td>
<td></td>
</tr>
</tbody>
</table>

702.4 Finite Element Classes

### 702.4.1 Stress, Strain and Elastoplastic State Classes

The next step in our development was to use the nDArray tool classes for constitutive level computations. The simple extension was design and implementation of infinitesimal stress and strain tensor classes, namely stresstensor and straintensor. Both classes are quite similar, they inherit all the functions from
the tensor class and we add some tools that are specific to them. Both stress and strain tensors are implemented as full second order $3 \times 3$ tensors. Symmetry of stress and strain tensor was not used to save storage space. Table 702.5 summarizes some of the main functions added on for the stress tensor class.

<table>
<thead>
<tr>
<th>operator or function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iinvariant1()</td>
<td>first stress invariant I1</td>
</tr>
<tr>
<td>Iinvariant2()</td>
<td>second stress invariant I2</td>
</tr>
<tr>
<td>Iinvariant3()</td>
<td>third stress invariant I3</td>
</tr>
<tr>
<td>Jinvariant2()</td>
<td>second deviatoric stress invariant J2</td>
</tr>
<tr>
<td>Jinvariant3()</td>
<td>third deviatoric stress invariant J3</td>
</tr>
<tr>
<td>deviator()</td>
<td>stress deviator</td>
</tr>
<tr>
<td>principal()</td>
<td>principal stresses on diagonal</td>
</tr>
<tr>
<td>sigma_octahedral()</td>
<td>octahedral mean stress</td>
</tr>
<tr>
<td>tau_octahedral()</td>
<td>octahedral shear stress</td>
</tr>
<tr>
<td>xi()</td>
<td>Haigh–Westergard coordinate $\xi$</td>
</tr>
<tr>
<td>rho()</td>
<td>Haigh–Westergard coordinate $\rho$</td>
</tr>
<tr>
<td>p_hydrostatic()</td>
<td>hydrostatic stress invariant</td>
</tr>
<tr>
<td>q_deviatoric()</td>
<td>deviatoric stress invariant</td>
</tr>
<tr>
<td>theta()</td>
<td>$\theta$ stress invariant (Lode’s angle)</td>
</tr>
</tbody>
</table>

Table 702.5: Additional methods for stress tensor class.

Further on, we defined an elastoplastic state, which according to incremental theory of elastoplasticity with internal variables, is completely defined with the stress tensor and a set of internal variables. This definition led us to define an elastoplastic state termed class ep_state. Objects of type ep_state contain a stress tensor and a set of scalar or tensorial internal variables.

### 702.4.2 Material Model Classes

With all the previous developments, the design and implementation of various elastoplastic material models was not a difficult task. A generic class Material_Model defines techniques that form a framework for small deformation elastoplastic computations. Table 702.6 summarizes some of the main methods defined for the Material_Model class in terms of yield ($F$) and potential ($Q$) functions.

---

2 Internal variables can be characterized as tensors of even order, where, for example, zero tensor is a scalar internal variable associated with isotropic hardening and second order tensors can be associated with kinematic hardening.  

---
<table>
<thead>
<tr>
<th>operator or function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>( F ) Yield function value</td>
</tr>
<tr>
<td>( dF/\sigma_{ij} )</td>
<td>( \partial F/\partial \sigma_{ij} )</td>
</tr>
<tr>
<td>( dQ/\sigma_{ij} )</td>
<td>( \partial Q/\partial \sigma_{ij} )</td>
</tr>
<tr>
<td>( d^2Q/\sigma_{ij}\sigma_{kl} )</td>
<td>( \partial^2 Q/\partial \sigma_{ij}\partial \sigma_{kl} )</td>
</tr>
<tr>
<td>( dp/\sigma_{ij} )</td>
<td>( \partial p/\partial \sigma_{ij} )</td>
</tr>
<tr>
<td>( dq/\sigma_{ij} )</td>
<td>( \partial q/\partial \sigma_{ij} )</td>
</tr>
<tr>
<td>( d\theta/\sigma_{ij} )</td>
<td>( \partial \theta/\partial \sigma_{ij} )</td>
</tr>
<tr>
<td>( d^2p/\sigma_{ij}\sigma_{kl} )</td>
<td>( \partial^2 p/\partial \sigma_{ij}\partial \sigma_{kl} )</td>
</tr>
<tr>
<td>( d^2q/\sigma_{ij}\sigma_{kl} )</td>
<td>( \partial^2 q/\partial \sigma_{ij}\partial \sigma_{kl} )</td>
</tr>
<tr>
<td>( d^2\theta/\sigma_{ij}\sigma_{kl} )</td>
<td>( \partial^2 \theta/\partial \sigma_{ij}\partial \sigma_{kl} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForwardPredictorEPState</td>
<td>Explicit predictor elastoplastic state</td>
</tr>
<tr>
<td>BackwardEulerEPState</td>
<td>Implicit return elastoplastic state</td>
</tr>
<tr>
<td>ForwardEulerEPState</td>
<td>Explicit return elastoplastic state</td>
</tr>
<tr>
<td>BackwardEulerCTensor</td>
<td>Algorithmic tangent stiffness tensor</td>
</tr>
<tr>
<td>ForwardEulerCTensor</td>
<td>Continuum tangent stiffness tensor</td>
</tr>
</tbody>
</table>

Table 702.6: Some of the methods in material model class.

It is important to note that all the material model dependent functions are defined as virtual functions. Integration algorithms are designed and implemented using template algorithms, and each implemented material model appends its own yield and potential functions and appropriate derivatives. Implementation of additional material models requires coding of yield and potential functions and respective derivative functions.

### 702.4.3 Stiffness Matrix Class

Starting from the incremental equilibrium of the stationary body, the principle of virtual displacements and with the finite element approximation of the displacement field \( u \approx \bar{u}_a = H_I \bar{u}_a \), the weak form of equilibrium can be expressed as (Zienkiewicz and Taylor, 1991a)

\[
\bigcup_m \int_{V_m} \bar{H}_{I,b} E_{abcd} \bar{H}_{J,d} dV_m \bar{\bar{u}}_J = \bigcup_m \int_{V_m} f_a \bar{H}_I dV_m \quad \text{or} \quad (f_{Ia}(\bar{u}_J))_{\text{int}} = \lambda (f_{Ia})_{\text{ext}}
\]
where $E_{abcd}$ is the constitutive tangent stiffness tensor$^3$. The element stiffness tensor is recognized as

$$k^e_{aIcJ} = \int_{V_m} H_{I,b} \tan E_{abcd} H_{J,d} dV^m$$

This generic form for the finite element stiffness tensor is easily programmed with the help of the nDarray tool. A simple implementation example is provided later. It should be noted that the element stiffness tensor in this case is a four–dimensional tensor. It is the task of the assembly function to collect proper terms for addition in a global stiffness matrix.

### 702.5 Examples

#### 702.5.1 Tensor Examples

Some of the basic tensorial calculations with tensors are presented. Tensors have a default constructor that creates a first order tensor with one element initialized to 0.0:

```cpp
    tensor t1;
```

Tensors can be constructed and initialized from a given set of numbers:

```cpp
    static double t2values[] = { 1,2,3, 4,5,6, 7,8,9 };
    tensor t2( 2, DefDim2, t2values); // order 2; 3x3 tensor (like matrix)
```

Here, DefDim2, DefDim3 and DefDim4 are arrays of dimensions for the second, third and fourth order tensor$^4$. A fourth order tensor with 0.0 value assignment and dimension 3 in each order ($3 \times 3 \times 3 \times 3$) is constructed in the following way:

```cpp
    tensor ZERO(4,DefDim4,0.0);
```

Tensors can be multiplied using indicial notation. The following example will do a tensorial multiplication of previously defined tensors $t2$ and $t4$ so that $tstl = t2_{ij}t4_{ijkl}t4_{klpq}t2_{pq}$. Note that the memory is dynamically allocated to accept the proper tensor dimensions that will result from the multiplication$^5$

```cpp
    tensor tstl = t2("ij")*t4("ijkl")*t4("klpq")*t2("pq");
```

Inversion of tensors is possible. It is defined for 2 and 4 order tensors only. The fourth order tensor inversion is done by converting it to matrix, inverting that matrix and finally converting matrix back to tensor.

```cpp
    tensor t4inv_2 = t4.inverse();
```

$^3$Which may be continuum or algorithmic (Jeremić and Sture, 1997) tangent stiffness tensor

$^4$In this case dimensions are 3 in every order.

$^5$In this case it will be zero dimensional tensor with one element.
There are two built-in tensor types, *Levi-Civita permutation tensor* $\epsilon_{ijk}$ and *Kronecker delta tensor* $\delta_{ij}$

```cpp
tensor e("e", 3, DefDim3); // Levi-Civita permutation tensor
tensor I2("I", 2, DefDim2); // Kronecker delta tensor
```

Trace and determinant functions for tensors are used as follows

```cpp
double deltatrace = I2.trace();
double deltadet = I2.determinant();
```

Tensors can be compared to within a square root of *machine epsilon* tolerance

```cpp
tensor I2again = I2;
if ( I2again == I2 )
    printf("I2again == I2 TRUE (OK)\n");
else
    printf("I2again == I2 NOT TRUE\n");
```

### 702.5.2 Fourth Order Isotropic Tensors

Some of the fourth order tensors used in continuum mechanics are built quite readily. The most general representation of the fourth order isotropic tensor includes the following fourth order unit isotropic tensors

```cpp
tensor I_{ijkl} = I2("ij")*I2("kl");
```

The resulting tensor $I_{ijkl}$ will have the correct indices, $I_{ijkl}I_{ijkl} = I2_{ij}I2_{kl}$. Note that $I_{ijkl}$ is just a name for the tensor, and the $_{ijkl}$ part reminds the implementor what that tensor is representing.

The real indices, $^{*}_{ijkl}$ in this case, are stored in the tensor object, and can be used further or changed appropriately. The next tensor that is needed is a fourth order unit tensor obtained by transposing the previous one in the minor indices,

```cpp
tensor I_{ikjl} = I_{ijkl}.transpose0110();
```

while the third tensor needed for representation of general isotropic tensor is constructed by using similar transpose function

```cpp
tensor I_{iljk} = I_{ijkl}.transpose0111();
```

The inversion function can be checked for fourth order tensors:

---

6Machine epsilon ($\text{macheps}$) is defined as the smallest distinguishable positive number (in a given precision, i.e. float (32 bits), double (64 bits) or long double (80 bits), such that $1.0 + \text{macheps} > 1.0$ yields true on the given computer platform. For example, double precision arithmetics (64 bits), on the Intel 80x86 platform yields $\text{macheps}=1.08E-19$ while on the Sun SPARCstation and DEC platforms $\text{macheps}=2.22E-16$.

7Remember that $I2$ was constructed as the Kronecker delta tensor $\delta_{ij}$.
tensor I_ikjl_inv_2 = I_ikjl.inverse();
if ( I_ikjl == I_ikjl_inv_2 )
    printf(" I_ikjl == I_ikjl_inv_2 (OK) !");
else
    printf(" I_ikjl != I_ikjl_inv_2 !");

Creating a symmetric and skew symmetric unit fourth order tensors gets to be quite simple by using
tensor addition and scalar multiplication

    tensor I4s = (1./2.)*(I_ikjl+I_iljk);
    tensor I4sk = (1./2.)*(I_ikjl-I_iljk);

Another interesting example is a numerical check of the $e - \delta$ identity (Lubliner, 1990)

\[
e_{ijm}e_{klm} = \delta_{ik}\delta_{jl} - \delta_{il}\delta_{jk}
\]

    tensor id = e("ijm")*e("klm") - (I_ikjl - I_iljk);
if ( id == ZERO )
    printf(" e-delta identity HOLDS !! ");

702.5.3 Elastic Isotropic Stiffness and Compliance Tensors

The linear isotropic elasticity tensor $E_{ijkl}$ can be built from Young's modulus $E$ and Poisson's ratio $\nu$

\[
de E_{ijkl} = \frac{2\cdot Ey\cdot nu}{2\cdot(1+nu)\cdot(1-2\cdot nu)}\cdot I_{ijkl} + \frac{Ey}{1+nu}\cdot I_{4s};
\]

Similarly, the compliance tensor is

\[
de D_{ijkl} = \frac{-nu/Ey}{I_{ijkl}} + \frac{(1.0+nu)/Ey}{I_{4s}};
\]

One can multiply the two and check if the result is equal to the symmetric fourth order unit tensor

\[
de test = E("ijkl")*D("klpq");
if ( test == I_{4s} )
    printf(" test == I_{4s} TRUE (OK up to sqrt(macheps)) ");
else
    printf(" test == I_{4s} NOTTRUE ");

The linear isotropic elasticity and compliance tensors can be obtained in a different way, by using Lamé
constants $\lambda$ and $\mu$

\[
de \lambda = nu * Ey / (1. + nu) / (1. - 2. * nu);
de \mu = Ey / (2. * (1. + nu));
de E = lambda*I_{ijkl} + (2.*mu)*I_{4s}; // stiffness tensor
de D = (-nu/Ey)*I_{ijkl} + (1.)/(2.*mu)*I_{4s}; // compliance tensor
\]
702.5.4 Second Derivative of $\theta$ Stress Invariant

As an extended example of \texttt{nDarray} tool usage, the implementation for the second derivative of the stress invariant $\theta$ (Lode angle) is presented. The derivative is used for implicit constitutive integration schemes applied to three invariant material models. The original equation reads:

\[
\frac{\partial^2 \theta}{\partial \sigma_{pq} \partial \sigma_{mn}} =
\begin{align*}
&- \left( \frac{9}{2} \frac{\cos 3\theta}{q^4 \sin (3\theta)} + \frac{27}{4} \frac{\cos 3\theta}{q^4 \sin^3 3\theta} \right) s_{pq} s_{mn} + \frac{81}{4} \frac{1}{q^4 \sin^3 3\theta} s_{pq} t_{mn} + \\
&+ \left( \frac{81}{4} \frac{1}{q^4 \sin 3\theta} + \frac{81}{4} \frac{\cos^2 3\theta}{q^4 \sin^3 3\theta} \right) t_{pq} s_{mn} - \frac{243}{4} \frac{\cos 3\theta}{q^4 \sin^3 3\theta} t_{pq} t_{mn} + \\
&+ \frac{3}{2} \frac{\cos (3\theta)}{q^2 \sin (3\theta)} p_{pqmn} - \frac{9}{2} \frac{1}{q^3 \sin (3\theta)} w_{pqmn}
\end{align*}
\]

where:

\[
q = \sqrt{\frac{3}{2}} s_{ij} s_{ij} ; \quad \cos 3\theta = \frac{3\sqrt{3}}{2} \frac{1}{\sqrt{(\frac{1}{2} s_{ij} s_{ij})^3}} ; \quad s_{ij} = \sigma_{ij} - \frac{1}{3} \sigma_{kk} \delta_{ij}
\]

\[
w_{pqmn} = s_{np} \delta_{qm} + s_{qm} \delta_{np} - \frac{2}{3} s_{qp} \delta_{nm} - \frac{2}{3} \delta_{pq} s_{mn} ; \quad p_{pqmn} = \delta_{mp} \delta_{nq} - \frac{1}{3} \delta_{pq} \delta_{mn}
\]

and the implementation follows:

```cpp
tensor Yield_Criteria::d2thetaoverds2(strestensor & stress)
{
    tensor ret( 4, DefDim4, 0.0);
    tensor I2("I", 2, DefDim2);
    tensor I_pqmn = I2("pq")*I2("mn");
    tensor I_pmqn = I_pqmn.transpose0110();
    double J2D = stress.Jinvariant2();
    tensor s = stress.deviator();
    tensor t = s("qk")*s("kp") - I2*(J2D*(2.0/3.0));
    double theta = stress.theta();
    double q_dev = stress.q_deviatoric();
    //setting up some constants
    double c3t = cos(3*theta);
    double s3t = sin(3*theta);
    double s3t3 = s3t*s3t*s3t;
    double q3 = q_dev * q_dev * q_dev;
```
double q4 = q3 * q_dev;
double q5 = q4 * q_dev;
double q6 = q5 * q_dev;
double tempss = -(9.0/2.0)*(c3t)/(q4*s3t)-(27.0/4.0)*(c3t/(s3t3*q4));
double tempst = +(81.0/4.0)*(1.0)/(s3t3*q5);
double temps = +(81.0/4.0)*(1.0)/(s3t3*q5);
double temptt = -(243.0/4.0)*(c3t/(s3t3*q6));
double tempw = -(9.0/2.0)*(1.0/(s3t*q3));
tensor s_pq_d_mn = s("pq")*I2("mn");
tensor s_pn_d_mq = s_pq_d_mn.transpose0101();
tensor d_pq_s_mn = I2("pq")*s("mn");
tensor d_pn_s_mq = d_pq_s_mn.transpose0101();
tensor p = I_pmqn - I_pqmn*(1.0/3.0);
tensor w = s_pn_d_mq+d_pn_s_mq - s_pq_d_mn*(2.0/3.0)-d_pq_s_mn*(2.0/3.0);
// finally
ret = (s("pq")*s("mn")*tempss + s("pq")*t("mn")*tempst +
t("pq")*s("mn")*temps + t("pq")*t("mn")*temptt +
p*tempw + w*tempw );
return ret;

702.5.5 Application to Computations in Elastoplasticity

A useful application of the previously described classes is for elastoplastic computations. If the Newton iterative scheme is used at the global equilibrium level, then in order to preserve a quadratic rate, a consistent, algorithmic tangent stiffness (ATS) tensor should be used. For a general class of three–invariant, non–associated, hardening or softening material models, ATS is defined (Jeremić and Sture, 1997) as:

\[
\text{cons}^{ep}_{pqmn} = R_{pqmn} - \frac{R_{pqkl} n^+ H_{kl} n^+ h_{ij} R_{ijmn}}{n^+ n_{ot} R_{otpq} n^+ H_{pq} + n^+ \xi_s h_s}
\]

where

\[
m_{kl} = \frac{\partial Q}{\partial \sigma_{kl}} ; \quad n_{kl} = \frac{\partial F}{\partial \sigma_{kl}} ; \quad \xi_s = \frac{\partial F}{\partial q_s} ; \quad T_{ijmn} = \delta_{im} \delta_{nj} + \Delta \lambda E_{ijkl} \frac{\partial m_{kl}}{\partial \sigma_{mn}}
\]

\[
H_{kl} = n^+ h_{kl} + \Delta \lambda \frac{\partial m_{kl}}{\partial q_s} h_s ; \quad R_{mnkl} = (n^+ T_{ijmn})^{-1} E_{ijkl}
\]

A straightforward implementation of the above tensorial formula follows:

```c++
    double Ey = Criterion.E();
```
double nu = Criterion.nu();
tensor Eel = StiffnessTensorE(Ey,nu);
tensor I2("I", 2, DefDim2);
tensor I_ijkl = I2("ij")*I2("kl");
tensor I_ikjl = I_ijkl.transpose0110();
tensor m = Criterion.dQods(final_stress);
tensor n = Criterion.dFods(final_stress);
double lambda = current_lambda_get();
tensor d2Qoverds2 = Criterion.d2Qods2(final_stress);
tensor T = I_ikjl + Eel("ijkl")*d2Qoverds2("klmn")*lambda;
tensor Tinv = T.inverse();
tensor R = Tinv("ijmn")*Eel("ijkl");
double h_ = h(final_stress);
double xi_ = xi(final_stress);
double hardMod_ = h_ * xi_;
tensor d2Qodqast2 = d2Qoverdqast2(final_stress);
tensor H = m + d2Qodqast2 * lambda * h_;

//
tensor upper = R("pqkl")*H("kl")*n("ij")*R("ijmm");
double lower = (n("ot")*R("otpq")*H("pq")).trace();
lower = lower + hardMod_;
tensor Ep = upper*(1./lower);
tensor Eep = R - Ep; // elastoplastic ATS constitutive tensor

This ATS tensor can be used further in building finite element stiffness tensors, as will be shown in our next example.

702.5.6 Stiffness Matrix Example

By applying a numerical integration technique to the stiffness matrix equation

\[ k_{el}^{ab} = \int_{V_m} H_{Ib} E_{abcd} H_{Jd} dV_m \]

individual contributions are summed into the element stiffness tensor. This process can be implemented on a integration point level by using the \texttt{nDarray} tool as

\[ K = K + H("Ib") * E("abcd") * H("Jd") * weight; \]

It is interesting to note the lack of loops at this level of implementation. However, there exists a loop over integration points which contributes stiffness to the element tensor.
702.6 Performance Issues

In the course of developing the nDarray tool, execution speed was not a priority or issue that we tried to perfect. The benefit of being able to implement and test various numerical algorithms in a straightforward manner was the main concern. The efficiency of the nDarray tool when compared with FORTRAN or C was never assessed. In all honesty, some of the formulae implemented in C++ with the help of the nDarray tool would be difficult to implement in FORTRAN or C. The entire question of efficiency of the nDarray as compared to FORTRAN or C codes might thus remain unanswered for the time being.

The efficiency of C++ for numerical computations has been under consideration (Robison, 1996) for some time now. Poor efficiency and possible remedies for improving efficiency of C++ computations has been reported in literature (Robison, 1996) (Veldhuizen, 1995b) (Veldhuizen, 1996). Novel techniques, such as Template Expressions (Veldhuizen, 1995b) can be used to achieve and sometimes surpass the performance of hand-tuned FORTRAN or C codes.

702.7 Summary and Future Directions

A novel programming tool, named nDarray, has been presented which facilitates implementation of tensorial formulae. It was shown how OOP and efficient programming in C++ allows building of new concrete data types, in this case tensors of any order. In a number of examples these new data types were shown to be useful in implementing tensorial formulae associated with the numerical solution of various elastic and elastoplastic problems with the finite element method. The nDarray tool is been used in developing of the FEMtools tools library. The FEMtools tools library includes a set of finite elements, various solvers, solution procedures for non-linear finite element system of equations and other useful functions.
Appendix 703

Closed Form Gradients to the Plastic Potential Function

(1993-1994-)

2718
703.1 Chapter Summary and Highlights

A complete derivation of gradients to the Potential and Yield function follows. The yield function $F$ and potential function $Q$ are functions of the stress tensor $\sigma_{ij}$ and plastic internal variable tensor $q_*$. Only derivatives with respect to the stress tensor $\sigma_{ij}$ are given here. It is assumed that any stress state can be represented with three stress invariants $p, q$ and $\theta$ given in the following form:

$$p = -\frac{1}{3}I_1 \quad q = \sqrt{3J_{2D}} \quad \cos 3\theta = \frac{3\sqrt{3}}{2} \frac{J_{3D}}{\sqrt{(J_{2D})^3}}$$ (703.1)

$$I_1 = \sigma_{kk} \quad J_{2D} = \frac{1}{2}s_{ij}s_{ij} \quad J_{3D} = \frac{1}{3}s_{ij}s_{jk}s_{ki} \quad s_{ij} = \sigma_{ij} - \frac{1}{3}\sigma_{kk}\delta_{ij}$$ (703.2)

and stresses are chosen as positive in tension. One can write the Potential Function in the following form:

$$Q = Q(p, q, \theta)$$ (703.3)

and the derivation follows. Hopefully the pace of derivation is rather slow, thus little explanation will be given until the end of the derivation. Chain rule of differentiation yields:

$$\frac{\partial Q}{\partial \sigma_{ij}} = \frac{\partial Q}{\partial p} \frac{\partial p}{\partial \sigma_{ij}} + \frac{\partial Q}{\partial q} \frac{\partial q}{\partial \sigma_{ij}} + \frac{\partial Q}{\partial \theta} \frac{\partial \theta}{\partial \sigma_{ij}}$$ (703.4)

and the intermediate derivatives are:

$$\frac{\partial p}{\partial \sigma_{ij}} = \frac{\partial (-\frac{1}{3}\sigma_{kk})}{\partial \sigma_{ij}} = -\frac{1}{3}\delta_{ij}$$ (703.5)

$$\frac{\partial q}{\partial \sigma_{ij}} = \frac{\partial \sqrt{3J_{2D}}}{\partial \sigma_{ij}} = \frac{\sqrt{3}}{2} \frac{1}{\sqrt{J_{2D}}} \frac{\partial J_{2D}}{\partial \sigma_{ij}} = \frac{\sqrt{3}}{2} \frac{1}{\sqrt{J_{2D}}} \frac{\partial J_{2D}}{\partial \sigma_{ij}} = \frac{3}{2} \frac{1}{q} s_{ij}$$ (703.6)
\[
\frac{\partial \theta}{\partial \sigma_{ij}} = \frac{1}{3} \frac{-1}{\sqrt{1 - \left(\frac{3\sqrt{3} J_3}{J_2^{3/2}}\right)^2}} \frac{3\sqrt{3}}{2} \left( \frac{\partial J_3 D}{\partial \sigma_{ij}} \frac{1}{\sqrt{(J_2 D)^3}} - \frac{3}{2} J_3 \frac{\partial J_2 D}{\partial \sigma_{ij}} \frac{1}{\sqrt{(J_2 D)^5}} \right) = \\
\frac{1}{3} \frac{3\sqrt{3}}{2} \frac{-t_{ij}}{\sqrt{(J_2 D)^3}} \frac{1}{\sqrt{(J_2 D)^3}} + \frac{3}{2} J_3 s_{ij} \frac{1}{\sqrt{(J_2 D)^5}} \right) = \\
\frac{1}{\sin 3\theta} \frac{\sqrt{3}}{2} \left( \frac{3}{2} J_3 D \frac{1}{\sqrt{(J_2 D)^5}} s_{ij} - \frac{1}{\sqrt{(J_2 D)^3}} t_{ij} \right) = \\
\frac{\sqrt{3}}{2} \frac{1}{\sin (3\theta)} \left( \frac{\sqrt{3} \cos (3\theta)}{q^2} s_{ij} - \frac{3\sqrt{3}}{q^3} t_{ij} \right) = \\
\frac{3 \cos (3\theta)}{2 q^2 \sin (3\theta)} s_{ij} - \frac{9}{2 q^3 \sin (3\theta)} t_{ij} \right)
\]

Second derivatives of the potential function \(Q\) using again the chain rule of differentiation are as follows:
\[
\frac{\partial^2 Q}{\partial \sigma_{pq} \partial \sigma_{mn}} = \frac{\partial}{\partial \sigma_{mn}} \left( \frac{\partial Q}{\partial \sigma_{pq}} \right) = \frac{\partial}{\partial \sigma_{mn}} \left( \frac{\partial Q}{\partial \sigma_{pq}} \frac{\partial p}{\partial \sigma_{pq}} + \frac{\partial Q}{\partial \sigma_{pq}} \frac{\partial q}{\partial \sigma_{pq}} + \frac{\partial Q}{\partial \sigma_{pq}} \frac{\partial \theta}{\partial \sigma_{pq}} \right) = 0
\]

and the intermediate derivatives are as follows:

\[
\frac{\partial^2 p}{\partial \sigma_{pq} \partial \sigma_{mn}} = \frac{\partial^2}{\partial \sigma_{pq} \partial \sigma_{mn}} \left( -\frac{1}{3} \sigma_{kk} \right) = \frac{\partial}{\partial \sigma_{mn}} \left( -\frac{1}{3} \delta_{kp} \delta_{qk} \right) = 0
\]

\[
\frac{\partial^2 q}{\partial \sigma_{pq} \partial \sigma_{mn}} = \frac{\partial}{\partial \sigma_{mn}} \left( \frac{\partial q}{\partial \sigma_{pq}} \right) = 0
\]

\[
\frac{\partial}{\partial \sigma_{mn}} \left( \frac{\sqrt{3}}{2} \frac{1}{\sqrt{J_{2D}}} s_{pq} \right) = \frac{\sqrt{3}}{2} \frac{1}{\sqrt{J_{2D}}} \frac{\partial s_{pq}}{\partial \sigma_{mn}} + \frac{\sqrt{3}}{2} \frac{\partial}{\partial \sigma_{mn}} \frac{1}{\sqrt{J_{2D}}} s_{pq} = 0
\]

\[
\frac{\sqrt{3}}{2} \frac{1}{\sqrt{J_{2D}}} \left( \delta_{pm} \delta_{nq} - \frac{1}{3} \delta_{pq} \delta_{km} \delta_{nk} \right) = \frac{\sqrt{3}}{2} \frac{1}{\sqrt{J_{2D}}} \left( \frac{1}{\sqrt{J_{2D}}} \right)^3 s_{mn} s_{pq}
\]

\[
\frac{3 \frac{1}{4} q}{\frac{1}{3} \delta_{pq} \delta_{mn}} - \frac{1}{9} \frac{1}{4} q^3 s_{mn} s_{pq}
\]
Let us introduce a slightly different form for the equation \( \frac{\partial^2 \theta}{\partial \sigma_{pq} \partial \sigma_{mn}} \) in order to simplify writing:

\[
\frac{\partial \theta}{\partial \sigma_{pq}} = \frac{3}{2} \frac{\cos (3\theta)}{q^2 \sin (3\theta)} s_{pq} - \frac{9}{2} \frac{1}{q^3 \sin (3\theta)} t_{pq} = \]

\[
= AS \; s_{pq} + AT \; t_{pq}
\]

where:

\[
AS = \frac{3}{2} \frac{\cos (3\theta)}{q^2 \sin (3\theta)}
\]

\[
AT = -\frac{9}{2} \frac{1}{q^3 \sin (3\theta)}
\]

Now the problem will be separated in two smaller problems, namely:

\[
\frac{\partial^2 \theta}{\partial \sigma_{pq} \partial \sigma_{mn}} = \frac{\partial \theta}{\partial \sigma_{pq}} = \frac{\partial (\frac{3}{2} \frac{\cos (3\theta)}{q^2 \sin (3\theta)} s_{pq} - \frac{9}{2} \frac{1}{q^3 \sin (3\theta)} t_{pq})}{\partial \sigma_{mn}} = \frac{\partial (AS \; s_{pq} + AT \; t_{pq})}{\partial \sigma_{mn}} = \frac{\partial (AS \; s_{pq})}{\partial \sigma_{mn}} + \frac{\partial (AT \; t_{pq})}{\partial \sigma_{mn}}
\]

Now let us take a look at \( \frac{\partial (AS \; s_{pq})}{\partial \sigma_{mn}} \). Since:
\[
\frac{\partial (AS \ s_{pq})}{\partial \sigma_{mn}} =
\]

\[
\frac{\partial AS}{\partial \sigma_{mn}} \ s_{pq} + AS \ \frac{\partial s_{pq}}{\partial \sigma_{mn}} =
\]

\[
\left( \frac{\partial AS}{\partial q} \ \frac{\partial q}{\partial \sigma_{mn}} + \frac{\partial AS}{\partial \theta} \ \frac{\partial \theta}{\partial \sigma_{mn}} \right) \ s_{pq} + AS \ \frac{\partial s_{pq}}{\partial \sigma_{mn}} =
\]

\[
\left( \frac{-3 \ \cot(3\theta)}{q^3} \ \frac{3}{2} q \ s_{mn} +
\frac{-4.5 \ \csc(3\theta)^2}{q^2} \left( \frac{3}{2} q^2 \sin(3\theta) \ s_{mn} - \frac{9}{2} q^3 \sin(3\theta) \ t_{mn} \right) \right) \ s_{pq} +
\frac{3 \ \cos(3\theta)}{2 \ q^2 \sin(3\theta)} \ p_{pqmn} =
\]

\[
\frac{-9}{2 \ q^4 \sin(3\theta)} \ s_{pq} \ s_{mn} - \frac{27 \ \cos 3\theta}{4 \ q^4 \sin^3 3\theta} \ s_{pq} \ s_{mn} + \frac{81}{4 \ q^3 \sin^3 3\theta} \ s_{pq} \ t_{mn} +
\frac{3 \ \cos(3\theta)}{2 \ q^2 \sin(3\theta)} \ p_{pqmn} =
\]

\[
\frac{-\left( \frac{9 \ \cos 3\theta}{2 \ q^4 \sin(3\theta)} + \frac{27 \ \cos 3\theta}{4 \ q^4 \sin^3 3\theta} \right) \ s_{pq} \ s_{mn} + \frac{81}{4 \ q^3 \sin^3 3\theta} \ s_{pq} \ t_{mn} +
\frac{3 \ \cos(3\theta)}{2 \ q^2 \sin(3\theta)} \ p_{pqmn} =
\]

where:

\[ p_{pqmn} = \frac{\partial s_{pq}}{\partial \sigma_{mn}} = \left( \delta_{mp} \delta_{nq} - \frac{1}{3} \delta_{pq} \delta_{mn} \right) \]

is the projection tensor and:

\[ \frac{\partial AS}{\partial q} = \frac{-3 \ \cot(3\theta)}{q^3} \]
\[
\frac{\partial AS}{\partial \theta} = -\frac{4.5 \csc(3\theta)^2}{q^2}
\]

The second member is \( \frac{\partial (AT \ t_{pq})}{\partial \sigma_{mn}} \):

\[
\frac{\partial (AT \ t_{pq})}{\partial \sigma_{mn}} = \frac{\partial AT}{\partial \sigma_{mn}} t_{pq} + AT \frac{\partial t_{pq}}{\partial \sigma_{mn}} = \\
\left( \frac{\partial AT}{\partial q} \frac{\partial q}{\partial \sigma_{mn}} + AT \frac{\partial AT}{\partial \theta} \frac{\partial \theta}{\partial \sigma_{mn}} \right) t_{pq} + AT \frac{\partial t_{pq}}{\partial \sigma_{mn}} = \\
13.5 \cot(3\theta) \csc(3\theta) \left( \frac{3}{q^4} \cos(3\theta) \frac{1}{2 q^2 \sin(3\theta)} s_{mn} - \frac{9}{2 q^3 \sin(3\theta)} t_{mn} \right) t_{pq} + \\
- \frac{9}{2 q^3 \sin(3\theta)} w_{pqmn} = \\
\frac{81}{4} \frac{1}{q^5 \sin 3\theta} t_{pq} s_{mn} + \frac{81}{4} \frac{\cos^2 3\theta}{q^5 \sin^3 3\theta} t_{pq} s_{mn} - \frac{243}{4} \frac{\cos 3\theta}{q^6 \sin^3 3\theta} t_{pq} t_{mn} - \\
- \frac{9}{2 q^3 \sin(3\theta)} w_{pqmn} = \\
\frac{81}{4} \frac{1}{q^5 \sin 3\theta} + \frac{81}{4} \frac{\cos^2 3\theta}{q^5 \sin^3 3\theta} \right) t_{pq} s_{mn} - \frac{243}{4} \frac{\cos 3\theta}{q^6 \sin^3 3\theta} t_{pq} t_{mn} - \\
- \frac{9}{2 q^3 \sin(3\theta)} w_{pqmn}
\]

where:

\[
w_{pqmn} = \frac{\partial t_{pq}}{\partial \sigma_{mn}} = s_{np} \delta_{qm} + s_{qm} \delta_{np} - \frac{2}{3} s_{qp} \delta_{nm} - \frac{2}{3} \delta_{pq} s_{mn}
\]

\[
\frac{\partial AT}{\partial q} = \frac{13.5 \csc(3\theta)}{q^4}
\]
\[
\frac{\partial AT}{\partial \theta} = \frac{13.5 \cot(3\theta) \csc(3\theta)}{q^3}
\]

Then finally by collecting terms back again we have:

\[
\frac{\partial^2 \theta}{\partial \sigma_{pq} \partial \sigma_{mn}} = \frac{\partial (AS \ s_{pq})}{\partial \sigma_{mn}} + \frac{\partial (AT \ t_{pq})}{\partial \sigma_{mn}} =
\]

\[
- \left( \frac{9}{2} \frac{\cos 3\theta}{q^4 \sin (3\theta)} + \frac{27}{4} \frac{\cos 3\theta}{q^4 \sin^3 3\theta} \right) s_{pq} s_{mn} + \frac{81}{4} \frac{1}{q^5 \sin^3 3\theta} s_{pq} t_{mn} +
\]

\[
+ \left( \frac{81}{4} \frac{1}{q^5 \sin 3\theta} + \frac{81}{4} \frac{\cos^2 3\theta}{q^5 \sin^3 3\theta} \right) t_{pq} s_{mn} - \frac{243}{4} \frac{\cos 3\theta}{q^6 \sin^3 3\theta} t_{pq} t_{mn} +
\]

\[
+ \frac{3}{2} \frac{\cos (3\theta)}{q^2 \sin (3\theta)} p_{pqmn} - \frac{9}{2} \frac{1}{q^3 \sin (3\theta)} w_{pqmn}
\]
Appendix 704

Hyperelasticity, Detailed Derivations

(1995-1996-)}
704.1 Chapter Summary and Highlights

704.2 Simo–Serrin’s Formula

In order to derive the analytical gradient of the fourth order tensor
\[ M_{IJKL} = \frac{\partial M_{IJ}}{\partial C_{KL}} \]  
we shall proceed by using the third equation in (106.86).

\[
\frac{\partial M_{IJ}}{\partial C_{KL}} = \frac{1}{D(A)} \left( I_{IKJL} - \frac{\partial I_1}{\partial C_{KL}} \delta_{IJ} + 2\lambda(A) \frac{\partial \lambda(A)}{\partial C_{KL}} \delta_{IJ} + \frac{\partial I_3}{\partial C_{KL}} \lambda^{-2}_{(A)} (C^{-1})_{IJ} - 2\lambda(A) \frac{\partial \lambda(A)}{\partial C_{KL}} I_3 (C^{-1})_{IJ} + \frac{\partial (C^{-1})_{IJ}}{\partial C_{KL}} \lambda^{-2}_{(A)} I_3 \right) - \frac{1}{D(A)} \frac{\partial D(A)}{\partial C_{KL}} \left( C_{IJ} - (I_1 - \lambda^2_{(A)}) \delta_{IJ} + I_3 \lambda^{-2}_{(A)} (C^{-1})_{IJ} \right)
\]  
where it was used that
\[ \frac{\partial C_{IJ}}{\partial C_{KL}} = I_{IKJL} \]  

Derivatives \( \frac{\partial \lambda(A)}{\partial C_{KL}} \) can be found by starting from equation for \( C_{IJ} \) (106.63) and differentiating it
\[ dC_{IJ} = 2\lambda_A d\lambda(A) \left( N^{(A)} I^{(A)} \right)_A + \lambda^2_A \left( dN^{(A)} N^{(A)} \right)_A + \lambda^2_A \left( \lambda(A) dN^{(A)} \right)_A \]  
By premultiplying previous equation with \( N^{(A)} I^{(A)} \) and post-multiplying with \( N^{(A)} I^{(A)} \), and by noting that
\[ N^{(A)} I^{(A)} \equiv 0 \quad ; \quad \| N^{(A)} \| = 1 \]  
we get
\[ N^{(A)} dC_{IJ} N^{(A)} = 2\lambda_A d\lambda(A) \]  
or
\[ dC_{IJ} N^{(A)} I^{(A)} = dC_{IJ} \lambda(A) M^{(A)} = 2\lambda_A d\lambda(A) \Rightarrow \frac{\partial \lambda(A)}{\partial C_{KL}} = \frac{1}{2} \lambda(A) (M^{(A)})_{KL} \]  
It can be proved\(^1\) that
\[ \frac{\partial I_1}{\partial C_{KL}} = \delta_{IJ} \quad ; \quad \frac{\partial I_2}{\partial C_{KL}} = I_1 \delta_{KL} - C_{KL} \quad ; \quad \frac{\partial I_3}{\partial C_{KL}} = I_3 (C^{-1})_{KL} \]  

\(^1\)See Marsden and Hughes (1983)
and since $I_3 = J^2$

$$\frac{\partial J}{\partial C_{KL}} = \frac{1}{2} J (C^{-1})_{KL}$$  \hspace{1cm} (704.9)

With this in mind, equation (704.2) can be rewritten as:

$$\frac{\partial M_{IJ}}{\partial C_{KL}} = \frac{1}{D_{(A)}} \left[ I_{IKJL} - \delta_{KL} \delta_{IJ} + 2\lambda_{(A)}^2 \frac{1}{2} M_{KL}^{(A)} \delta_{IJ} + 
+ I_3 \lambda_{(A)}^{-2} (C^{-1})_{IJ} (C^{-1})_{KL} - \lambda_{(A)}^{-2} I_3 (C^{-1})_{IJ} M_{KL}^{(A)} + 
+ \frac{1}{2} ((C^{-1})_{IK}(C^{-1})_{JL} + (C^{-1})_{IL}(C^{-1})_{JK}) \lambda_{(A)}^{-2} I_3 \right] - 
- \frac{1}{D_{(A)}} \frac{\partial D_{(A)}}{\partial C_{KL}} M_{IJ}$$  \hspace{1cm} (704.10)

where the definition of $M_{IJ}$ from equation (106.86) was used and also:

$$\frac{\partial (C^{-1})_{IJ}}{\partial C_{KL}} = -\frac{1}{2} \left( ((C^{-1})_{IK}(C^{-1})_{JL} + (C^{-1})_{IL}(C^{-1})_{JK}) = I_{IJKL}^{(C^{-1})} \right)$$  \hspace{1cm} (704.11)

Relation (704.11) can be obtained if one starts from the identity:

$$C_{IJ} (C^{-1})_{JK} = \delta_{IK}$$  \hspace{1cm} (704.12)

which after differentiation reads:

$$dC_{IJ} (C^{-1})_{JK} + C_{IJ} d(C^{-1})_{JK} = 0 \Rightarrow 
\Rightarrow d(C^{-1})_{JK} = -(C^{-1})_{JM} dC_{MN} (C^{-1})_{NK} = 
= -\frac{1}{2} \left( ((C^{-1})_{JM}(C^{-1})_{KN} + (C^{-1})_{JN}(C^{-1})_{KM}) \right) dC_{MN} \Rightarrow 
\Rightarrow \frac{\partial (C^{-1})_{JK}}{\partial C_{MN}} = -\frac{1}{2} \left( ((C^{-1})_{JM}(C^{-1})_{KN} + (C^{-1})_{JN}(C^{-1})_{KM}) \right)$$  \hspace{1cm} (704.13)

The derivative of $D_{(A)}$, that was defined in equation (106.77) as

$$D_{(A)} = 2\lambda_{(A)}^4 - I_1 \lambda_{(A)}^2 + I_3 \lambda_{(A)}^{-2}$$  \hspace{1cm} (704.14)
is given by:

\[
\frac{\partial D(A)}{\partial C_{KL}} = 8\lambda^3(A) \frac{\partial \lambda(A)}{\partial C_{KL}} - \frac{\partial I_1}{\partial C_{KL}} \lambda^2(A) - 2\lambda(A) I_1 \frac{\partial \lambda(A)}{\partial C_{KL}} + \frac{\partial I_3}{\partial C_{KL}} \lambda^{-2}(A) - 2\lambda^{-3}(A) I_3 \frac{\partial \lambda(A)}{\partial C_{KL}}
\]

\[
= 4\lambda(A) M_{KL}^{(A)} - \delta_{KL} \lambda^2(A) - \lambda^2(A) I_1 M_{KL}^{(A)} + I_3 (C^{-1})_{KL} \lambda^{-2}(A) - \lambda_{(A)}^{-2} I_3 M_{KL}^{(A)}
\]

\[
= \left(4\lambda(A) - \lambda^2(A) I_1 - \lambda^{-2}(A) I_3\right) M_{KL}^{(A)} - \delta_{KL} \lambda^2(A) + I_3 (C^{-1})_{KL} \lambda^{-2}(A)
\]

\[
= D'(A) M_{KL}^{(A)} - \delta_{KL} \lambda^2(A) + I_3 (C^{-1})_{KL} \lambda^{-2}(A)
\]

(704.15)

where \( D'(A) = 4\lambda^4(A) - \lambda^2(A) I_1 - \lambda^{-2}(A) I_3 \). With the previous derivations, equation (704.10) can be written in expanded form as:

\[
\frac{\partial M_{IJ}}{\partial C_{KL}} = \frac{1}{D(A)} \left( I_{KJL} - \delta_{KL} \delta_{IJ} + 2\lambda^2(A) \frac{1}{2} M_{KL}^{(A)} \delta_{IJ} + I_3 \lambda^2(A) (C^{-1})_{IJ} (C^{-1})_{KL} - \lambda^{-2}(A) I_3 (C^{-1})_{IJ} M_{KL}^{(A)} + \frac{1}{2} ((C^{-1})_{IK}(C^{-1})_{JL} + (C^{-1})_{IL}(C^{-1})_{JK}) \lambda^{-2}(A) I_3 - \left(D'(A) M_{KL}^{(A)} - \delta_{KL} \lambda^2(A) + I_3 (C^{-1})_{KL} \lambda^{-2}(A)\right) M_{IJ}\right)
\]

(704.16)

If one collects similar terms, equation (704.16), also known as Simo–Serrin’s formula can be written in the final form as:

\[
\frac{\partial M_{IJ}}{\partial C_{KL}} = M_{IJKL} = \frac{1}{D(A)} \left(I_{KJL} - \delta_{KL} \delta_{IJ} + \lambda^2(A) \left(\delta_{IJ} M_{KL}^{(A)} + M_{IJ}^{(A)} \delta_{KL}\right) + I_3 \lambda^2(A) \left((C^{-1})_{IJ} (C^{-1})_{KL} + \frac{1}{2} ((C^{-1})_{IK}(C^{-1})_{JL} + (C^{-1})_{IL}(C^{-1})_{JK})\right) - \lambda^{-2}(A) I_3 \left((C^{-1})_{IJ} M_{KL}^{(A)} + M_{IJ}^{(A)} (C^{-1})_{KL}\right) - D'(A) M_{IJ}^{(A)} M_{KL}^{(A)}\right)
\]

(704.17)

### 704.3 Derivation of \(\partial^{2\text{vol}} W / (\partial C_{IJ} \partial C_{KL})\)

The volumetric part \(\partial^{2\text{vol}} W / (\partial C_{IJ} \partial C_{KL})\) can be derived by starting from the equation (106.96):

---

Jeremić et al.
University of California, Davis
where equations (704.11) and (704.9) were used.

704.4 Derivation of $\partial^{2\text{iso}}W/(\partial C_{IJ} \partial C_{KL})$

The isochoric part $\partial^{2\text{iso}}W/(\partial C_{IJ} \partial C_{KL})$ can be derived by starting from equation (106.97)
\[
\frac{\partial^2 \text{iso} W(\lambda_A)}{\partial C_{IJ} \partial C_{KL}} = \frac{1}{2} \left( w_A \left( M_{IJ}^{(A)} \right)_A \right) \partial C_{KL} = \frac{1}{2} \frac{\partial w_A}{\partial C_{KL}} (M_{IJ}^{(A)})_A + \frac{1}{2} w_A \frac{\partial (M_{IJ}^{(A)})_A}{\partial C_{KL}} = \frac{1}{2} \frac{\partial w_A}{\partial C_{KL}} \frac{\partial \lambda_B}{\partial C_{KL}} (M_{IJ}^{(A)})_A + \frac{1}{2} w_A (M_{IJKL}^{(A)})_A = \frac{1}{4} Y_{AB} (M_{KL}^{(B)})_B (M_{IJ}^{(A)})_A + \frac{1}{2} w_A (M_{IJKL}^{(A)})_A
\] (704.19)

where equation (704.7) was used and tensor \( Y_{AB} \) is defined as:

\[
Y_{AB} = \frac{\partial w_A}{\partial \lambda_B} \lambda(B)
\] (704.20)

---

### 704.5 Derivation of \( w_A \)

\[
w_A = \frac{\partial \text{iso} W}{\partial \lambda_A} = \frac{\partial \text{iso} W}{\partial \lambda_B} \frac{\partial \lambda_B}{\partial \lambda_A} \lambda_A
\] (704.21)

where \( \lambda_B \) is the isochoric part of the stretch defined as

\[
\lambda_B = J^{-\frac{1}{2}} \lambda_B
\] (704.22)
Derivation of $\tilde{\lambda}_B$ in equation (704.22) it follows

$$\frac{\partial \tilde{\lambda}_B}{\partial \lambda_{(A)}} = \frac{\partial J^{-\frac{1}{2}}}{\partial \lambda_{(A)}} \lambda_B + J^{-\frac{1}{2}} \frac{\partial \lambda_B}{\partial \lambda_{(A)}} = -\frac{1}{3} J^{-\frac{1}{2}} \lambda^{-1}_{(A)} \lambda_B + J^{-\frac{1}{2}} \delta_{B(A)}$$

(704.23)

since

$$\frac{\partial J^{-\frac{1}{2}}}{\partial \lambda_{(A)}} = -\frac{1}{3} J^{-\frac{1}{2}} \frac{\partial \lambda_1 \lambda_2 \lambda_3}{\partial \lambda_{(A)}} = -\frac{1}{3} J^{-\frac{1}{2}} J \lambda^{-1}_{(A)} = -\frac{1}{3} J^{-\frac{1}{2}} \lambda^{-1}_{(A)}$$

(704.24)

then

$$w_A = -\frac{1}{3} J^{-\frac{1}{2}} \frac{\partial^{iso}W}{\partial \lambda_B} \lambda^{-1}_{(A)} \lambda_B + J^{-\frac{1}{2}} \frac{\partial^{iso}W}{\partial \lambda_B} \lambda_{B(A)} = -\frac{1}{3} \frac{\partial^{iso}W}{\partial \lambda_B} \tilde{\lambda}_B + \frac{\partial^{iso}W}{\partial \lambda_{(A)}} \tilde{\lambda}_{(A)}$$

(704.25)

### 704.6 Derivation of $Y_{AB}$

By starting from equation 704.20

$$Y_{AB} = \frac{\partial w_A}{\partial \lambda_B} \lambda_{(B)}$$

(704.26)

and by using equation 704.25

$$w_A = -\frac{1}{3} \frac{\partial^{iso}W}{\partial \lambda_C} \tilde{\lambda}_C + \frac{\partial^{iso}W}{\partial \lambda_{(A)}} \tilde{\lambda}_{(A)}$$

(704.27)

we can write:

$$Y_{AB} = \frac{\partial w_A}{\partial \lambda_D} \frac{\partial \tilde{\lambda}_D}{\partial \lambda_B} \lambda_{(B)}$$

(704.28)

By first considering $\partial w_A/\partial \tilde{\lambda}_D$ we get:

$$\frac{\partial w_A}{\partial \tilde{\lambda}_D} = \partial \left( -\frac{1}{3} \frac{\partial^{iso}W}{\partial \lambda_C} \tilde{\lambda}_C + \frac{\partial^{iso}W}{\partial \lambda_{(A)}} \tilde{\lambda}_{(A)} \right)$$

$$= -\frac{1}{3} \frac{\partial^{2iso}W}{\partial \lambda_C \partial \lambda_D} \tilde{\lambda}_C - \frac{1}{3} \frac{\partial^{2iso}W}{\partial \lambda_C \partial \lambda_D} \tilde{\lambda}_C + \frac{\partial^{2iso}W}{\partial \lambda_{(A)} \partial \lambda_D} \tilde{\lambda}_{(A)} + \frac{\partial^{iso}W}{\partial \lambda_{(A)}} \frac{\partial \tilde{\lambda}_{(A)}}{\partial \lambda_D}$$

$$= -\frac{1}{3} \frac{\partial^{2iso}W}{\partial \lambda_C \partial \lambda_D} \tilde{\lambda}_C - \frac{1}{3} \frac{\partial^{2iso}W}{\partial \lambda_C \partial \lambda_D} \delta_{CD} + \frac{\partial^{2iso}W}{\partial \lambda_{(A)} \partial \lambda_D} \tilde{\lambda}_{(A)} + \frac{\partial^{iso}W}{\partial \lambda_{(A)}} \delta_{(A)D}$$

(704.29)
Next, from equation 704.23, we have that

$$\frac{\partial \lambda_D}{\partial \lambda(B)} = -\frac{1}{3} J^{-\frac{1}{2}} \lambda(B) \lambda_D + J^{-\frac{1}{2}} \delta(D(B)) \tag{704.30}$$

and by multiplying the result for $\partial w_A / \partial \lambda_D$ from equation 704.29 and the result for $\partial \lambda_D / \partial \lambda(B)$ from equation 704.30 we obtain:

$$\frac{\partial w_A}{\partial \lambda_D} \frac{\partial \tilde{\lambda}_D}{\partial \lambda(B)} = + \frac{1}{9} \frac{\partial^{2\text{iso}}W}{\partial \lambda_C \partial \lambda_D} \tilde{\lambda}_C J^{-\frac{1}{2}} \lambda(B) \lambda_D - \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda_D} \tilde{\lambda}_C J^{-\frac{1}{2}} \delta(D(B))
+ \frac{1}{9} \frac{\partial^{2\text{iso}}W}{\partial \lambda_D} J^{-\frac{1}{2}} \lambda(B) \lambda_D
- \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda(A)} \partial \lambda_D \tilde{\lambda}_A J^{-\frac{1}{2}} \lambda(B) \lambda_D
+ \frac{\partial^{2\text{iso}}W}{\partial \lambda(D)} \tilde{\lambda}_A J^{-\frac{1}{2}} \delta(D(B))
- \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda(A)} \lambda(B) \partial \lambda_D \tilde{\lambda}_A J^{-\frac{1}{2}} \lambda(B)
+ \frac{\partial^{2\text{iso}}W}{\partial \lambda(A)} \tilde{\lambda}(A) J^{-\frac{1}{2}} \delta(D(B))
- \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda(A)} \lambda(B) \tilde{\lambda}(A) J^{-\frac{1}{2}} \delta(D(B)) \tag{704.31}$$

where equation 704.22 was used. The final form for $Y_{AB}$ is obtained by multiplying equation 704.31 with $\tilde{\lambda}(B)$ to obtain:

$$Y_{AB} = \frac{\partial w_A}{\partial \lambda_D} \frac{\partial \tilde{\lambda}_D}{\partial \lambda(B)} \lambda(B) =
+ \frac{1}{9} \frac{\partial^{2\text{iso}}W}{\partial \lambda_C \partial \lambda_D} \tilde{\lambda}_C \lambda(B) \tilde{\lambda}_D \lambda(B) - \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda_C \partial \lambda(D)} \tilde{\lambda}_C \tilde{\lambda}(B)
+ \frac{1}{9} \frac{\partial^{2\text{iso}}W}{\partial \lambda_D} \lambda(B) \tilde{\lambda}_D \lambda(B) - \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda(D)} \tilde{\lambda}(B)
- \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda(A)} \tilde{\lambda}(A) \lambda(B) \tilde{\lambda}_D \lambda(B) + \frac{\partial^{2\text{iso}}W}{\partial \lambda(A) \partial \lambda(D)} \tilde{\lambda}(A) \tilde{\lambda}(B)
- \frac{1}{3} \frac{\partial^{2\text{iso}}W}{\partial \lambda(A)} \lambda(B) \tilde{\lambda}(A) \lambda(B) + \frac{\partial^{2\text{iso}}W}{\partial \lambda(A)} \tilde{\lambda}(A) \tilde{\lambda}(B) \tag{704.32}$$

By recognizing that $\lambda(B) \lambda(B) = 1$ and after rearranging elements, we can finally write the equation for $Y_{AB}$ as:
\[ Y_{AB} = \frac{\partial^{iso}W}{\partial \lambda_{(A)}} \delta_{(A)(B)} \hat{\lambda}_{(B)} + \frac{\partial^{2iso}W}{\partial \lambda_{(A)} \partial \lambda_{(B)}} \hat{\lambda}_{(A)} \hat{\lambda}_{(B)} \]
\[ - \frac{1}{3} \left( \frac{\partial^{2iso}W}{\partial \lambda_{C} \partial \lambda_{(B)}} \hat{\lambda}_{C} \hat{\lambda}_{(B)} + \frac{\partial^{iso}W}{\partial \lambda_{(B)}} \hat{\lambda}_{(B)} + \frac{\partial^{2iso}W}{\partial \lambda_{(A)} \partial \lambda_{D}} \hat{\lambda}_{(A)} \hat{\lambda}_{D} + \frac{\partial^{iso}W}{\partial \lambda_{(A)}} \hat{\lambda}_{(A)} \right) \]
\[ + \frac{1}{9} \left( \frac{\partial^{2iso}W}{\partial \lambda_{C} \partial \lambda_{D}} \hat{\lambda}_{C} \hat{\lambda}_{D} + \frac{\partial^{iso}W}{\partial \lambda_{D}} \hat{\lambda}_{D} \right) \]  (704.33)
Appendix 705

Body and Surface Wave Analytic Solutions

(In collaboration with Dr. Nima Tafazzoli, Mr. Chang-Gyun Jeong, and Dr. Hexiang Wang)
705.1 3D Seismic Wave Field: Analytic Solution

In this chapter, wave field generation methods using analytic solution, and frequency wavenumber integration method are introduced. Theoretical background and examples are presented for each method.

705.1.1 Analytic solution

Seismic waves can be categorized as body waves and surface waves. The seismic body waves are traveling through the interior of the earth whereas the surface waves are traveling through the surface of the earth.

There are two different body waves, such as the pressure wave (also called as P wave, Figure 705.1 top) and the shear waves (also called as S wave, Figure 705.1 bottom). The shear waves which have the same velocity $V_S$ can be polarized along its plane location to the direction of propagation (vertical plane - SV and horizontal plane - SH, Figure 705.1 bottom).

Surface waves are mainly categorized as Rayleigh waves (Rayleigh, 1885) and Love waves (Love,
The Rayleigh waves induce elliptical ground movement near the surface (Figure 705.2) whereas the Love waves induce shearing movement (Figure 705.3).

Following sections introduce full three dimensional exact-solution and its examples for plane body and surface waves in homogeneous media. The chapter does not include full derivation of equations since it’s beyond the scope. Instead, final equations for body and surface waves are presented.

The original works for those problems are done by Green (1848), Knott (1899), and Wiechert and Zoeppritz (1907). The notations and equations hereafter are mainly based on Semblat and Pecker (2009) and Aki and Richard’s work (Aki and Richards, 2002).

### 705.1.1.1 Wave equations for body waves

Hereafter, the reflection and refraction coefficients are indicated by using its wave component symbols. An acute and grave accents are adapted to explain the direction of propagation. The acute accent indicates an upcoming wave, and the grave accent indicates a down-going wave (e.g. $\dot{P}$, $\ddot{P}$). For example, if the upcoming incident wave type is $P$ and down-going reflected wave type is $S$, then reflected wave will be indicated as $\dot{P}\ddot{S}$.

**Reflected and transmitted waves arising from incident SH wave** In the case of SH incident wave on the interface between two half-spaces, reflected wave is SH wave (Figure 705.4). The vector
Figure 705.4: Schematic cartoon to show all possible coefficients of reflection and transmission with SH incident wave displacements for the downgoing and upgoing incident SH waves can be calculated as below equation (705.1) and (705.2), respectively (Aki and Richards, 2002).

\[
\begin{align*}
(\text{Downgoing SH}) &= S(0, S, 0) \exp \left[ i \omega \left( px + \frac{\cos j_1}{\beta_1} z - t \right) \right] \\
(\text{Upgoing SH}) &= S(0, S, 0) \hat{S} \hat{S} \exp \left[ i \omega \left( px - \frac{\cos j_1}{\beta_1} z - t \right) \right] \\
(\text{Downgoing SH}) &= S(0, S, 0) \hat{S} \hat{S} \exp \left[ i \omega \left( px + \frac{\cos j_2}{\beta_2} z - t \right) \right] \\
(\text{Upgoing SH}) &= S(0, S, 0) \hat{S} \hat{S} \exp \left[ i \omega \left( px - \frac{\cos j_2}{\beta_2} z - t \right) \right] \\
(\text{downgoing SH}) &= S(0, S, 0) \hat{S} \hat{S} \exp \left[ i \omega \left( px + \frac{\cos j_1}{\beta_1} z - t \right) \right] \\
(\text{Upgoing SH}) &= S(0, S, 0) \hat{S} \hat{S} \exp \left[ i \omega \left( px - \frac{\cos j_1}{\beta_1} z - t \right) \right] \\
(\text{Downgoing SH}) &= S(0, S, 0) \hat{S} \hat{S} \exp \left[ i \omega \left( px + \frac{\cos j_2}{\beta_2} z - t \right) \right]
\end{align*}
\]

(705.1)

where,

\[
\begin{align*}
\dot{\hat{S}} \hat{S} &= \frac{\rho_1 \beta_1 \cos j_1 - \rho_2 \beta_2 \cos j_2}{\Delta} \\
\hat{S} \dot{\hat{S}} &= \frac{2 \rho_2 \beta_2 \cos j_2}{\Delta} \\
\dot{\hat{S}} \dot{\hat{S}} &= \frac{2 \rho_1 \beta_1 \cos j_1}{\Delta} \\
\hat{S} \dot{S} &= -\dot{\hat{S}} \hat{S} \\
\Delta &= \rho_1 \beta_1 \cos j_1 + \rho_2 \beta_2 \cos j_2
\end{align*}
\]

(705.3)

where \(\alpha\) is the P wave velocity, \(\beta\) is the S wave velocity, \(\rho\) is the density, \(p = (\sin i)/\alpha = (\sin j)/\beta\) is the ray parameter, and \(S\) is the amplitude of the incident wave.

Figure 705.4 shows all possible reflection and transmission coefficients with incident SH waves. Equation (705.4) is a 'scattering matrix' which includes every possible reflection and transmission coefficients.
Figure 705.5: Schematic cartoon to show all possible coefficients of reflection and transmission with P/SV incident wave for the problem. The matrix components have one to one relation with Figure 705.4.

\[
\begin{pmatrix}
\hat{S}\hat{S} & \hat{S}\hat{S} \\
\hat{S}\hat{S} & \hat{S}\hat{S}
\end{pmatrix}
\]  

(705.4)

**Reflected and transmitted waves arising from incident P/SV wave**  The displacements generated by the downgoing incident P/SV and upgoing incident P/SV waves can be calculated as below equation (705.5), (705.6), (705.7), and (705.8), respectively (Figure 705.5) (Aki and Richards, 2002).

(Downgoing P) = \( S(\sin i_1, 0, \cos i_1) \exp \left[ i\omega \left( px + \frac{\cos i_1}{\alpha_1} z - t \right) \right] \)

(Upgoing P) = \( S(\sin i_1, 0, -\cos i_1) \hat{P}\hat{P} \exp \left[ i\omega \left( px - \frac{\cos i_1}{\alpha_1} z - t \right) \right] \)

(Upgoing SV) = \( S(\cos j_1, 0, \sin j_1) \hat{P}\hat{S} \exp \left[ i\omega \left( px - \frac{\cos j_1}{\beta_1} z - t \right) \right] \)

(Downgoing P) = \( S(\sin i_2, 0, \cos i_2) \hat{P}\hat{P} \exp \left[ i\omega \left( px + \frac{\cos i_2}{\alpha_2} z - t \right) \right] \)

(Downgoing SV) = \( S(\cos j_2, 0, -\sin j_2) \hat{P}\hat{S} \exp \left[ i\omega \left( px + \frac{\cos j_2}{\beta_2} z - t \right) \right] \)  

(705.5)
(Downgoing SV) = \( S(\cos j_1, 0, -\sin j_1) \exp \left[ i\omega \left( px + \frac{\cos j_1}{\beta_1} z - t \right) \right] \)

(Upgoing P) = \( S(\sin i_1, 0, -\cos i_1) \hat{S} \hat{P} \exp \left[ i\omega \left( px - \frac{\cos i_1}{\alpha_1} z - t \right) \right] \)

(Upgoing SV) = \( S(\cos j_1, 0, \sin j_1) \hat{S} \hat{S} \exp \left[ i\omega \left( px - \frac{\cos j_1}{\beta_1} z - t \right) \right] \)

(Downgoing P) = \( S(\sin i_2, 0, \cos i_2) \hat{S} \hat{P} \exp \left[ i\omega \left( px + \frac{\cos i_2}{\alpha_2} z - t \right) \right] \)

(Downgoing SV) = \( S(\cos j_2, 0, -\sin j_2) \hat{S} \hat{S} \exp \left[ i\omega \left( px + \frac{\cos j_2}{\beta_2} z - t \right) \right] \) (705.6)

(Upgoing P) = \( S(\sin i_2, 0, -\cos i_2) \hat{S} \hat{P} \exp \left[ i\omega \left( px - \frac{\cos i_2}{\alpha_2} z - t \right) \right] \)

(Upgoing P) = \( S(\sin i_1, 0, -\cos i_1) \hat{P} \hat{P} \exp \left[ i\omega \left( px - \frac{\cos i_1}{\alpha_1} z - t \right) \right] \)

(Upgoing SV) = \( S(\cos j_1, 0, \sin j_1) \hat{P} \hat{S} \exp \left[ i\omega \left( px - \frac{\cos j_1}{\beta_1} z - t \right) \right] \)

(Downgoing P) = \( S(\sin i_2, 0, \cos i_2) \hat{P} \hat{P} \exp \left[ i\omega \left( px + \frac{\cos i_2}{\alpha_2} z - t \right) \right] \)

(Downgoing SV) = \( S(\cos j_2, 0, -\sin j_2) \hat{P} \hat{S} \exp \left[ i\omega \left( px + \frac{\cos j_2}{\beta_2} z - t \right) \right] \) (705.7)

(Upgoing SV) = \( S(\cos j_2, 0, \sin j_2) \exp \left[ i\omega \left( px - \frac{\cos j_2}{\beta_2} z - t \right) \right] \)

(Upgoing P) = \( S(\sin i_1, 0, -\cos i_1) \hat{S} \hat{P} \exp \left[ i\omega \left( px - \frac{\cos i_1}{\alpha_1} z - t \right) \right] \)

(Upgoing SV) = \( S(\cos j_1, 0, \sin j_1) \hat{S} \hat{S} \exp \left[ i\omega \left( px - \frac{\cos j_1}{\beta_1} z - t \right) \right] \)

(Downgoing P) = \( S(\sin i_2, 0, \cos i_2) \hat{S} \hat{P} \exp \left[ i\omega \left( px + \frac{\cos i_2}{\alpha_2} z - t \right) \right] \)

(Downgoing SV) = \( S(\cos j_2, 0, -\sin j_2) \hat{S} \hat{S} \exp \left[ i\omega \left( px + \frac{\cos j_2}{\beta_2} z - t \right) \right] \) (705.8)
where,

\[ \dot{P} P = \left[ \left( b \frac{\cos i_1}{\alpha_1} - c \frac{\cos i_2}{\alpha_2} \right) F - \left( a + d \frac{\cos i_1 \cos j_2}{\alpha_1 \beta_2} \right) H p^2 \right] / D \]

\[ \dot{P} S = -2 \frac{\cos i_1}{\alpha_1} \left( b + c d \frac{\cos i_2 \cos j_2}{\alpha_2 \beta_2} \right) p / (\beta_1 D) \]

\[ \dot{P} \dot{P} = 2 \rho_1 \frac{\cos i_1}{\alpha_1} F / (\alpha_2 D) \]

\[ \dot{P} \dot{S} = 2 \rho_1 \frac{\cos i_1}{\alpha_1} H p / (\beta_2 D) \]

\[ \dot{S} P = -2 \frac{\cos j_1}{\beta_1} \left( b + c d \frac{\cos i_2 \cos j_2}{\alpha_2 \beta_2} \right) p / (\alpha_1 D) \]

\[ \dot{S} S = -\left[ \left( b \frac{\cos j_1}{\beta_1} - c \frac{\cos j_2}{\beta_2} \right) E - \left( a + d \frac{\cos i_2 \cos j_1}{\alpha_2 \beta_1} \right) G p^2 \right] / D \]

\[ \dot{S} P = -2 \rho_1 \frac{\cos j_1}{\beta_1} G p / (\alpha_2 D) \]

\[ \dot{S} \dot{S} = 2 \rho_1 \frac{\cos j_1}{\beta_1} E / (\beta_2 D) \]

\[ \dot{P} \dot{P} = 2 \rho_2 \frac{\cos i_2}{\alpha_2} F / (\alpha_2 D) \]

\[ \dot{P} \dot{S} = -2 \rho_2 \frac{\cos i_2}{\alpha_2} G p / (\alpha_2 D) \]

\[ \dot{P} \dot{P} = -\left[ \left( b \frac{\cos i_1}{\alpha_1} - c \frac{\cos i_2}{\alpha_2} \right) F + \left( a + d \frac{\cos i_2 \cos j_1}{\alpha_2 \beta_1} \right) G p^2 \right] / D \]

\[ \dot{P} \dot{S} = 2 \frac{\cos i_2}{\alpha_2} \left( a c + b d \frac{\cos i_1 \cos j_1}{\alpha_1 \beta_1} \right) p / (\beta_2 D) \]

\[ \dot{S} P = 2 \frac{\cos j_2}{\beta_2} H p \beta_1 / (\alpha_1 D) \]

\[ \dot{S} \dot{S} = 2 \frac{\cos j_2}{\beta_2} E \beta_1 / (\beta_2 D) \]

\[ \dot{S} P = 2 \frac{\cos j_2}{\beta_2} \left( a c + b d \frac{\cos i_1 \cos j_1}{\alpha_1 \beta_1} \right) p / (\alpha_2 D) \]

\[ \dot{S} \dot{S} = \left[ \left( b \frac{\cos j_1}{\beta_1} - c \frac{\cos j_2}{\beta_2} \right) E + \left( a + d \frac{\cos i_1 \cos j_2}{\alpha_1 \beta_2} \right) H p^2 \right] / D \] (705.9)

\[ a = \rho_2 (1 - 2 \beta_2^2 p^2) - \rho_1 (1 - 2 \beta_1^2 p^2) \]

\[ b = \rho_2 (1 - 2 \beta_2^2 p^2) + 2 \rho_1 \beta_1^2 p^2 \]

\[ c = \rho_1 (1 - 2 \beta_1^2 p^2) + 2 \rho - 2 \beta_2^2 p^2 \]

\[ d = 2 (\rho_2 \beta_2^2 - \rho_1 \beta_1^2) \] (705.10)
\[ E = \frac{b \cos i_1}{\alpha_1} + \frac{c \cos i_2}{\alpha_2} \]
\[ F = \frac{b \cos j_1}{\beta_1} + \frac{c \cos j_2}{\beta_2} \]
\[ G = a - d \frac{\cos i_1 \cos j_2}{\alpha_1 \beta_2} \]
\[ H = a - d \frac{\cos i_2 \cos j_1}{\alpha_2 \beta_1} \]
\[ D = EF + GHp^2 = \frac{(\text{det} \ M)}{(\alpha_1 \alpha_2 \beta_1 \beta_2)} \] (705.11)

\[ M = \begin{pmatrix}
-\alpha_1 p & -\cos j_1 & \alpha_2 p & \cos j_2 \\
\cos i_1 & -\beta_1 p & \cos i_2 & -\beta_2 p \\
2\rho_1 \beta_1^2 p \cos i_1 & \rho_1 \beta_1 (1 - 2\beta^2_1 p^2) & 2\rho_2 \beta_2^2 p \cos i_2 & \rho_2 \beta_2 (1 - 2\beta^2_2 p^2) \\
-\rho_1 \alpha_1 (1 - 2\beta^2_1 p^2) & 2\rho_1 \beta_1^2 p \cos j_1 & \rho_2 \alpha_2 (1 - 2\beta^2_2 p^2) & -2\rho_2 \beta_2^2 p \cos j_2
\end{pmatrix} \] (705.12)

where \( \alpha \) is the P wave velocity, \( \beta \) is the S wave velocity, \( \rho \) is the mass density, \( p = (\sin i)/\alpha = (\sin j)/\beta \) is the ray parameter, and \( S \) is the amplitude of the incident wave.

Similar as the incident SH wave case, Figure 705.5 shows all possible reflection and transmission coefficients. Below equation (705.13) is a ‘scattering matrix’ which includes every possible reflection and transmission coefficients for the problem. The matrix components have one to one relation with Figure 705.5.

\[ \begin{pmatrix}
\dot{\mathbf{P}} \cdot \dot{\mathbf{P}} & \dot{\mathbf{S}} \cdot \dot{\mathbf{P}} & \ddot{\mathbf{P}} \cdot \dot{\mathbf{P}} & \dddot{\mathbf{P}} \\
\mathbf{P} \cdot \mathbf{S} & \mathbf{S} \cdot \mathbf{S} & \mathbf{P} \cdot \mathbf{S} & \mathbf{S} \cdot \mathbf{S} \\
\dot{\mathbf{P}} \cdot \dot{\mathbf{P}} & \dot{\mathbf{S}} \cdot \dot{\mathbf{P}} & \ddot{\mathbf{P}} \cdot \dot{\mathbf{P}} & \dddot{\mathbf{P}} \\
\dot{\mathbf{P}} \cdot \mathbf{S} & \dot{\mathbf{S}} \cdot \mathbf{S} & \ddot{\mathbf{P}} \cdot \mathbf{S} & \dddot{\mathbf{S}}
\end{pmatrix} \] (705.13)

### 705.1.1.2 Wave equations for surface waves

**Surface wave with incident SH wave – Love wave** The displacements of surface wave due to the incident SH wave can be obtained by solving wave equations under free-surface condition (zero traction on surface). Below equation (705.14) shows the time history displacement with incident SH
wave (Semblat and Pecker, 2009).

\[ u_y^i = A_{SH} \exp \left( \frac{i\omega}{V_S} \left( x \sin \theta_{iSH} + z \cos \theta_{iSH} - V_St \right) \right) \]
\[ u_y^R = R_{SH} \exp \left( \frac{i\omega}{V_S} \left( x \sin \theta_{RSH} - z \cos \theta_{RSH} - V_St \right) \right) \]
\[ u_y = u_y^i + u_y^R = 2A_{SH} \cos \left( \frac{\omega z \cos \theta_{iSH}}{V_S} \right) \exp \left[ \frac{i\omega}{V_S} \left( x \sin \theta_{i} - V_St \right) \right] \tag{705.14} \]

As shown in equation (705.14) displacements of waves induced by incident SH waves can be calculated by summing incident and reflected waves on the ground surface. The particle movement of a Love wave is perpendicular to the propagation plane.

**Surface wave with incident P/SV wave – Rayleigh wave.** Equation (705.15) shows the displacements of surface wave due to the incident P/SV waves (Semblat and Pecker, 2009).

\[ u_x = \frac{i\omega}{V_R} A \left( e^{a_z} - \frac{2ab}{b^2 + \omega^2/V_R^2} e^{b_z} \right) \exp \left[ \frac{i\omega}{V_R} (x - V_R t) \right] \]
\[ u_z = aA \left( e^{a_z} - \frac{2\omega^2/V_R^2}{b^2 + \omega^2/V_R^2} e^{b_z} \right) \exp \left[ \frac{i\omega}{V_R} (x - V_R t) \right] \tag{705.15} \]

where, \( a^2 = \frac{\omega^2}{V_R^2} - \frac{\omega^2}{V_P^2} \), and \( b^2 = \frac{\omega^2}{V_R^2} - \frac{\omega^2}{V_S^2} \).

Imaginary term on \( u_x \) shows that the components has a 90° phase shift from \( u_z \). The particle movement of the wave is ellipses in x, z plane.

### 705.2 Matlab code – body wave solution

#### Listing 705.1: Example MATLAB code for surface waves

```matlab
1 % ==============================================================
2 % body.m: inclined wave propagation closed—form solution for layered ground
3 %
4 % ref. Waves and Vibrations in Soils (Semblat and Pecker)
5 %
6 % Chang—Gyun Jeong
7 % Last update: 05/24/2011
8 % ==============================================================
9
10 clear all; clc;
11
12 %% initial condition
13 % wave type 'P' / 'SV' / 'SH'
14 wave = 'P';
15
16 % time
17 t_low = 0; t_up = 1;
```

dt = 0.01;
t_no = (t_up - t_low) / dt;

% x and z
min_x = 0; max_x = 100;
min_z = 0; max_z = 100;
delta_x = 2; delta_z = 2;
x = min_x:delta_x:max_x;
x = x';
z = min_z:delta_z:max_z;
z = z';

% frequency
freq = 10;
omega = freq * 2 * pi;

% incident angle and amplitude
th_P_i = 20; A_P = 1;

th_SV_i = 20; A_SV = 1;

th_SH_i = 20; A_SH = 1;

% velocity and shear modulus
V_S1 = 1000; V_S2 = 300;

nu = 0.3;

% shear modulus
gamma = 20000; % (N/m^3)

V_P1 = V_S1 * sqrt((2 - 2 * nu) / (1 - 2 * nu));

V_P2 = V_S2 * sqrt((2 - 2 * nu) / (1 - 2 * nu));

G_1 = gamma * V_S1^2;

G_2 = gamma * V_S2^2;

X_S = V_S2 / V_S1;

X_1 = V_P1 / V_S1;

X_2 = V_P2 / V_S2;

%% calculation

switch upper(wave)
  case {'P'}
    th_P_P_r = th_P_i;
    th_P_P_t = asind(sind(th_P_i) * V_P2 / V_P1);
    th_SV_P_r = asind(sind(th_P_i) * V_S1 / V_P1);
    th_SV_P_t = asind(sind(th_P_i) * V_S2 / V_P1);
    temp_left = [-sind(th_P_i), -cosd(th_SV_P_r), sind(th_P_P_t), -cosd(th_SV_P_r);...]
sind(2 * th_P_i), X_1 * cosd(2 * th_SV_P_r), X_1 / X_2 * X_S...]
cosd(th_P_i), -sind(th_SV_P_r), cosd(th_P_P_t), sind(th_SV_P_r);...]
  case {'Sv'}
    temp_right = A_P * [sind(th_P_i); cosd(th_P_i); sind(2 * th_P_i); X_1 *...]
cosd(2 * th_SV_P_r)];
    temp = temp_left \ temp_right;

R_SV_P = temp(1, 1);
R_P_P = temp(2, 1);
T_SV_P = temp(3, 1);
T_P_P = temp(4, 1);

% initialize matrix
ux_P_i = zeros(max(size(z)), max(size(x)));
uz_P_i = zeros(max(size(z)), max(size(x)));  
ux_P_P_r = zeros(max(size(z)), max(size(x)));  
uz_P_P_r = zeros(max(size(z)), max(size(x)));  
ux_SV_P_r = zeros(max(size(z)), max(size(x)));  
uz_SV_P_r = zeros(max(size(z)), max(size(x)));  
ux_P_P_t = zeros(max(size(z)), max(size(x)));  
uz_P_P_t = zeros(max(size(z)), max(size(x)));  
uz_SV_P_t = zeros(max(size(z)), max(size(x)));  
uz_SV_P_t = zeros(max(size(z)), max(size(x)));  

% calculate incident, reflected, and transmitted wave
for t = t_low : dt : t_up
  for i = 1 : max(size(x))
    for j = 1 : max(size(z))
      % incident P
      ux_P_i(i, j) = real(A_P * sind(th_P_i) * (exp((lii * omega / V_P1) ...  
        * (x(i, 1) * sind(th_P_i) + z(j, 1) * cosd(th_P_i) - V_P1 * t))));
      uz_P_i(i, j) = real(A_P * cosd(th_P_i) * (exp((lii * omega / V_P1) ...  
        * (x(i, 1) * sind(th_P_i) + z(j, 1) * cosd(th_P_i) - V_P1 * t))));
      % reflected P
      ux_P_P_r(i, j) = real(R_P_P * sind(th_P_P_r) * (exp((lii * omega / V_P1) ...  
        * (x(i, 1) * sind(th_P_P_r) + z(j, 1) * cosd(th_P_P_r) - V_P1 * t))));
      uz_P_P_r(i, j) = real(R_P_P * cosd(th_P_P_r) * (exp((lii * omega / V_P1) ...  
        * (x(i, 1) * sind(th_P_P_r) + z(j, 1) * cosd(th_P_P_r) - V_P1 * t))));
      % reflected SV
      ux_SV_P_r(i, j) = real(R_SV_P * cosd(th_SV_P_r) * (exp((lii * omega / V_S1) ...  
        * (x(i, 1) * sind(th_SV_P_r) + z(j, 1) * cosd(th_SV_P_r) - V_S1 * t))));
      uz_SV_P_r(i, j) = real(R_SV_P * sind(th_SV_P_r) * (exp((lii * omega / V_S1) ...  
        * (x(i, 1) * sind(th_SV_P_r) + z(j, 1) * cosd(th_SV_P_r) - V_S1 * t))));
      % transmitted P
      ux_P_P_t(i, j) = real(T_P_P * sind(th_P_P_t) * (exp((lii * omega / V_P2) ...  
        * (x(i, 1) * sind(th_P_P_t) + z(j, 1) * cosd(th_P_P_t) - V_P2 * t))));
      uz_P_P_t(i, j) = real(T_P_P * cosd(th_P_P_t) * (exp((lii * omega / V_P2) ...  
        * (x(i, 1) * sind(th_P_P_t) + z(j, 1) * cosd(th_P_P_t) - V_P2 * t))));
      % transmitted SV
      ux_SV_P_t(i, j) = real(T_SV_P * cosd(th_SV_P_t) * (exp((lii * omega / V_S2) ...  
        * (x(i, 1) * sind(th_SV_P_t) + z(j, 1) * cosd(th_SV_P_t) - V_S2 * t))));
      uz_SV_P_t(i, j) = real(T_SV_P * sind(th_SV_P_t) * (exp((lii * omega / V_S2) ...  
        * (x(i, 1) * sind(th_SV_P_t) + z(j, 1) * cosd(th_SV_P_t) - V_S2 * t))));
  end
end

% plot
subplot(3,4,1)
surf(x, z, uz_P_i,'EdgeColor','none'); axis([0 100 0 100 0 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Incident P x comp.');
subplot(3,4,2)
surf(x, z, uz_P_P_r,'EdgeColor','none'); axis([0 100 0 100 0 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Incident P z comp.');
subplot(3,4,5)
surf(x, z, uz_P_P_t,'EdgeColor','none'); axis([0 100 0 100 0 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected P x comp.');
subplot(3,4,6)
surf(x, z, uz_SV_P_r,'EdgeColor','none'); axis([0 100 0 100 0 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected P z comp.');
subplot(3,4,7)
surf(x, z, uz_SV_P_t,'EdgeColor','none'); axis([0 100 0 100 0 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected SV x comp.');
```matlab
% angle and amplitude
th_P_SV_r = asind(sind(th_SV_i) * V_P1 / V_S1);
th_P_SV_t = asind(sind(th_SV_i) * V_P2 / V_S1);
th_SV_SV_r = th_SV_i;
th_SV_SV_t = asind(sind(th_SV_i) * V_S2 / V_S1);

temp_left = [cosd(th_SV_i), sind(th_P_SV_r), cosd(th_SV_SV_t), sind(th_P_SV_t), ...
sind(th_SV_i), -cosd(th_P_SV_r), -sind(th_SV_SV_t), cosd(th_P_SV_t), ...
-cosd(2 * th_SV_i), (-1 / X_1) * sind(2 * th_P_SV_r), X_S ... * cosd(2 * th_SV_SV_t), (-X_S * X_2) * sind(2 * th_P_SV_t), ...
-sind(2 * th_SV_i), X_1 * cosd(2 * th_SV_SV_r), -X_S ... * sin(2 * th_SV_SV_t), -X_2 * X_S * cosd(2 * th_SV_SV_t)];

temp_right = A_SV * [cosd(th_SV_i); -sind(th_SV_i); cosd(2 * th_SV_i); ...
-sind(2 * th_SV_i)];

temp = temp_left \ temp_right;

R_P_SV = temp(1, 1);
R_SV_SV = temp(2, 1);
T_P_SV = temp(3, 1);
T_SV_SV = temp(4, 1);

% initialize matrix
uz_SV_i = zeros(max(size(z)), max(size(x)));
ux_SV_i = zeros(max(size(z)), max(size(x)));
uz_P_SV_r = zeros(max(size(z)), max(size(x)));
ux_P_SV_r = zeros(max(size(z)), max(size(x)));
uz_P_SV_t = zeros(max(size(z)), max(size(x)));
ux_P_SV_t = zeros(max(size(z)), max(size(x)));
uz_SV_SV_r = zeros(max(size(z)), max(size(x)));
ux_SV_SV_r = zeros(max(size(z)), max(size(x)));
uz_SV_SV_t = zeros(max(size(z)), max(size(x)));
ux_SV_SV_t = zeros(max(size(z)), max(size(x)));

% calculate incident, reflected, and transmitted wave
for t = t_low : dt : t_up
    for i = 1 : max(size(z))
        for j = 1 : max(size(x))
            % incident SV
        end
    end
end
```

ux_SV_i(i, j) = \text{real}(A_{SV} \times \cosd(th_{SV_i}) \times \exp((i \times \omega / V_{S1}) \times (x(i, 1) \times \cosd(th_{SV_i}) + z(j, 1) \times \cosd(th_{SV_i} - V_{S1} \times t))));
ux_SV_i(i, j) = \text{real}(-A_{SV} \times \sind(th_{SV_i}) \times \exp((i \times \omega / V_{S1}) \times (x(i, 1) \times \sind(th_{SV_i}) + z(j, 1) \times \cosd(th_{SV_i} - V_{S1} \times t))));

% reflected P
ux_P_SV_r(i, j) = \text{real}(R_{PSV} \times \cosd(th_{PSV_r}) \times \exp((i \times \omega / V_P1) \times (x(i, 1) \times \cosd(th_{PSV_r}) - z(j, 1) \times \cosd(th_{PSV_r} - V_{P1} \times t))));
ux_P_SV_r(i, j) = \text{real}(R_{PSV} \times \sind(th_{PSV_r}) \times \exp((i \times \omega / V_P1) \times (x(i, 1) \times \sind(th_{PSV_r}) - z(j, 1) \times \cosd(th_{PSV_r} - V_{P1} \times t))));

% transmitted P
ux_P_SV_t(i, j) = \text{real}(T_{PSV} \times \cosd(th_{PSV_t}) \times \exp((i \times \omega / V_P2) \times (x(i, 1) \times \cosd(th_{PSV_t}) + z(j, 1) \times \cosd(th_{PSV_t} - V_{P2} \times t))));
ux_P_SV_t(i, j) = \text{real}(T_{PSV} \times \sind(th_{PSV_t}) \times \exp((i \times \omega / V_P2) \times (x(i, 1) \times \sind(th_{PSV_t}) + z(j, 1) \times \cosd(th_{PSV_t} - V_{P2} \times t))));

% transmitted SV
ux_SV_SV_t(i, j) = \text{real}(T_{SVSV} \times \sind(th_{SVSV_t}) \times \exp((i \times \omega / V_S1) \times (x(i, 1) \times \sind(th_{SVSV_t}) + z(j, 1) \times \cosd(th_{SVSV_t} - V_{S1} \times t))));
ux_SV_SV_t(i, j) = \text{real}(T_{SVSV} \times \sind(th_{SVSV_t}) \times \exp((i \times \omega / V_S1) \times (x(i, 1) \times \sind(th_{SVSV_t}) + z(j, 1) \times \cosd(th_{SVSV_t} - V_{S1} \times t))));

end
end

% plot
subplot(3,4,1)
surf(x, z, ux_SV_i,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Incident SV x comp.');

subplot(3,4,2)
surf(x, z, ux_P_SV_r,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Incident SV z comp.');

subplot(3,4,5)
surf(x, z, ux_P_SV_t,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected P z comp.');

subplot(3,4,6)
surf(x, z, ux_P_SV_r,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected P x comp.');

subplot(3,4,7)
surf(x, z, ux_P_SV_t,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected SV x comp.');

subplot(3,4,8)
surf(x, z, ux_P_SV_r,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected SV z comp.');

subplot(3,4,9)
surf(x, z, ux_P_SV_t,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Refracted P x comp.');

subplot(3,4,10)
```matlab
surf(x, z, uz_P_SV_t,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected P z comp.')

subplot(3,4,11)
surf(x, z, uz_SV_SV_t,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected SV x comp.')

subplot(3,4,12)
surf(x, z, uz_SV_SV_t,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected SV z comp. ')

drawnow
end

case {'SH'}
% angle and amplitude
th_SH_r = th_SH_i;
th_SH_t = asind(sin(th_SH_i) * V_S2 / V_S1);
R_SH = A_SH * (G_1 * V_S2 * cosd(th_SH_i) - G_2 * V_S1 * cosd(th_SH_t));
T_SH = A_SH * (G_1 * V_S2 * cosd(th_SH_i) + G_2 * V_S1 * cosd(th_SH_t));

% initialize matrix
uy_SH_i = zeros(max(size(z)), max(size(x)));
uy_SH_r = zeros(max(size(z)), max(size(x)));
uy_SH_t = zeros(max(size(z)), max(size(x)));

% calculate incident, reflected, and transmitted wave
for t = t_low : dt : t_up
    for i = 1 : max(size(z))
        for j = 1 : max(size(z))
            % incident SH
            uy_SH_i(i, j) = real(A_SH *(exp((1i * omega / V_S1) * (x(i, 1)...
            * sin(th_SH_i) + x(j, 1) * cosd(th_SH_i) - V_S1 * t))));
            % reflected SH
            uy_SH_r(i, j) = real(R_SH *(exp((1i * omega / V_S1) * (x(i, 1)...
            * sin(th_SH_r) - x(j, 1) * cosd(th_SH_r) - V_S1 * t))));
            % refracted SH
            uy_SH_t(i, j) = real(T_SH *(exp((1i * omega / V_S2) * (x(i, 1)...
            * sin(th_SH_t) + x(j, 1) * cosd(th_SH_t) - V_S2 * t))));
            % adding incident and reflected SH
            adding_SH(i, j) = (uy_SH_i(i, j) - uy_SH_r(i, j)) / 2;
        end
    end
end

subplot(3,1,1)
surf(x, z, uy_SH_i,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Incident')

subplot(3,1,2)
surf(x, z, uy_SH_r,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected')

subplot(3,1,3)
surf(x, z, uy_SH_t,'EdgeColor','none'); axis([0 100 0 100 -4 4])
xlabel('x direction'); ylabel('z direction'); view(3)
title('Reflected')

surf(x, z, adding_SH,'EdgeColor','none'); xlabel('x direction'); ylabel('z direction'); view(3)
```

---

Jeremić et al. University of California, Davis
338  drawnow
339  end
340  end
341  end
### 705.3 Matlab code – surface wave solution

Listing 705.2: Example MATLAB code for surface waves

```matlab
% ==============================================================
% surf.m: inclined wave propagation closed—form exact solution for surface waves
% % ref. Waves and Vibrations in Soils (Semblat and Pecker, 2009)
% % Chang—Gyun Jeong
% % Last update: 03/27/2011
% ==============================================================

clear all;
c;clc;

% coordinates to see waves
x_cord = 0;
z_cord = 0;

% time limit for time history
t_low = 0; t_up = 10; dt = 0.01;

% wave frequency and initial time for phase
freq = 10; % natural frequency of wave
omega = freq * 2 * pi; % omega

% incident angle and amplitude
A_P = 1; % incident amp. (P wave)
A_SH = 1; % incident amp. (SH wave)

% velocity and shear modulus (r is assumed to 20 kN/m³)

v = 0.3; % Poisson's ratio
V_S1 = 50; % Vs for layer 1
V_R = V_S1 * (0.862 + 1.14 * v) / (1 + v); % Rayleigh wave velocity
V_P1 = V_S1 * sqrt((2 - 2 * v) / (1 - 2 * v)); % Vp for layer 1

% calculate Rayleigh waves
a = -1i * abs(sqrt(omega^2 / V_R^2 - omega^2 / V_P1^2));
b = -1i * abs(sqrt(omega^2 / V_R^2 - omega^2 / V_S1^2));

fq_u_x_R = A_P * 1i * omega / V_R * ...
    (exp(a * z_cord) - ((2 * a + b / (b^2 + omega^2 / V_R^2)) + exp(b * z_cord))) ...
    * exp((1i * omega / V_R) * (x_cord - V_R * t));

fq_u_z_R = A_P * a * ...
    (exp(a * z_cord) - ((2 * omega^2 / V_R^2) / (b^2 + omega^2 / V_R^2) + exp(b * z_cord))) ...
    * exp((1i * omega / V_R) * (x_cord - V_R * t));

% calculate Love waves

fq_u_y_L = 2 * A_SH * cosd(omega * z_cord * cosd(th_SH_i) / V_S1) ...
    * (exp((1i * omega / V_S1) * (x_cord * sind(th_SH_i) - V_S1 * t)));

% calculate phase angle and amplitude of harmonic motion

time_ang_x = angle(fq_u_x_R); % phase angle (R wave, x)
time_abs_x = abs(fq_u_x_R); % amplitude (R wave, x)
time_ang_z = angle(fq_u_z_R); % phase angle (R wave, z)
time_abs_z = abs(fq_u_z_R); % amplitude (R wave, z)
time_ang_y = angle(fq_u_y_L); % phase angle (L wave, y)
time_abs_y = abs(fq_u_y_L); % amplitude angle (L wave, y)

% calculate time history Rayleigh wave
```
60  time = t_low : dt : t_up;
61  u_x_R = time_abs_x * cos(2*freq*pi*time+time_ang_x);
62  u_z_R = time_abs_z * cos(2*freq*pi*time+time_ang_z);
63  % calculate time history Love wave
64  u_y_L = time_abs_y * cos(2*freq*pi*time+time_ang_y);
65  % plot
66  subplot(1,4,1)
67  plot(u_x_R, u_z_R)
68  xlabel('x'); ylabel('z');
69  axis([-4 4 -4 4])
70  subplot(1,4,2)
71  plot(u_y_L, u_z_R)
72  xlabel('y'); ylabel('z');
73  axis([-4 4 -4 4])
74  subplot(1,4,3)
75  plot(u_x_R, u_y_L)
76  xlabel('x'); ylabel('y');
77  axis([-4 4 -4 4])
78  subplot(1,4,4)
79  plot3(u_x_R, u_y_L, u_z_R)
80  axis([-4 4 -4 4 -4 4])
81  xlabel('x'); ylabel('y'); zlabel('z');
705.4 Matlab code – Ricker wavelet as an input motion

Listing 705.3: Example MATLAB code for Ricker wavelet as an input motion

```matlab
% ====================================
% Ricker.m: Ricker wavelet propagation on 3D space closed—form exact solution
% %
% % ref. Waves and Vibrations in Soils (Semblat and Pecker, 2009)
% %
% % Chang—Gyun Jeong
% % Last update: 01. 05. 2012.
% % ====================================
% clf; clc; clear all;

%% Initial condition
% max frequency, amplitude, and input angle of ricker wavelet
f_max = 1;
amplitude = 0.005;
th = 40;

% amplitude ratio
% ar_PP = 1;
% arR = 1;

% coordinate for extracting waves
xxx = 0;
zzz = 900;

dx = 100;
xx_min = 0;
xx_max = xxx;

dz = 100;
zz_min = 0;
zz_max = zzz;

nx = xx_max / dx + 1;
zx = zz_max / dz + 1;

% Poisson's ratio and wave velocity of ground
v = 0.1;
Vs = 700;
Vp = Vs * sqrt((2 - 2 * v) / (1 - 2 * v));
Vr = Vs * (0.862 + 1.14 * v) / (1 + v);

% time step, peak time, max and min time
dt = 0.01;
t_min = 0;
t_max = 20;
t_peak = 4;

% parameters for plotting
t_min_pl = 0;
t_max_pl = 10;
d_min_pl = -0.02;
d_max_pl = 0.02;

f_min_pl = 0;
f_max_pl = 5;
f_amp_min_pl = 0;
f_amp_max_pl = 0.0006;
```
% amplitude and angle calculation for reducing computation time
thrp = asind(sin(th) * Vp / Vs);
ths = asind(sin(th) * Vs / Vp);
ss = sind(th) / Vs;
c = cosd(th) / Vs;
srrp = sind(thrp) / Vs;
ccpp = cosd(thrp) / Vs;
srp = sind(th) / Vs;
ccp = cosd(th) / Vs;
srs = sind(thrs) / Vs;
ccrs = cosd(thrs) / Vs;
as = amplitude * sind(th);
ac = amplitude * cosd(th);

% Ricker wavelet
k = 1;
for t = t_min:dt:t_max
    j = 1;
    for z = z_min:dz:z_max
        i = 1;
        for x = x_min:dx:x_max
            SV_i_x(i, j) = ac * ...
            ((1 - 2 * pi/2 * f_max^2 * (t - t_peak + x * ss + z * cc)^2) * ...
            exp(-pi/2 * f_max^2 * ...
            (t - t_peak + x * ss + z * cc)^2));
            SV_i_z(i, j) = ac * ...
            ((1 - 2 * pi/2 * f_max^2 * (t - t_peak + x * ss + z * cc)^2) * ...
            exp(-pi/2 * f_max^2 * ...
            (t - t_peak + x * ss + z * cc)^2));
            SVM_r_x(i, j) = ac * cosd(thrp) * ...
            ((1 - 2 * pi/2 * f_max^2 * (t - t_peak + x * ssrp - z * ccpp)^2) * ...
            exp(-pi/2 * f_max^2 * ...
            (t - t_peak + x * ssrp - z * ccpp)^2));
            SVM_r_z(i, j) = ac * cosd(thrp) * ...
            ((1 - 2 * pi/2 * f_max^2 * (t - t_peak + x * ssrp + z * ccpp)^2) * ...
            exp(-pi/2 * f_max^2 * ...
            (t - t_peak + x * ssrp + z * ccpp)^2));
            P_i_x(i, j) = ac * ...
            ((1 - 2 * pi/2 * f_max^2 * (t + 1.35 - t_peak + x * ssp + z * ccpp)^2) * ...
            exp(-pi/2 * f_max^2 * ...
            (t + 1.35 - t_peak + x * ssp + z * ccpp)^2));
            P_i_z(i, j) = ac * ...
            ((1 - 2 * pi/2 * f_max^2 * (t + 1.35 - t_peak + x * ssp + z * ccpp)^2) * ...
            exp(-pi/2 * f_max^2 * ...
            (t + 1.35 - t_peak + x * ssp + z * ccpp)^2));
            PP_r_x(i, j) = as * ...
            ((1 - 2 * pi/2 * f_max^2 * (t + 1.35 - t_peak + x * ssp - z * ccpp)^2) * ...
            exp(-pi/2 * f_max^2 * ...
            (t + 1.35 - t_peak + x * ssp - z * ccpp)^2));
            PP_r_z(i, j) = as * cosd(th) * ...
\[
(1 - 2 * \pi^2 * f_{\text{max}}^2 * (t + 1.35 - t_{\text{peak}} + x * \text{ssp} + z * \text{ccp})^2) * \ldots \\
\exp(-\pi^2 * f_{\text{max}}^2 * \ldots \\
(t + 1.35 - t_{\text{peak}} + x * \text{ssp} + z * \text{ccp})^2));
\]

\[
\text{PSV}\_r_x(i, j) = \text{ac} * \cos(\text{thrs}) * \ldots \\
((1 - 2 * \pi^2 * f_{\text{max}}^2 * (t + 1.35 - t_{\text{peak}} + x * \text{ssrs} - z * \text{ccrs})^2) * \ldots \\
\exp(-\pi^2 * f_{\text{max}}^2 * \ldots \\
(t + 1.35 - t_{\text{peak}} + x * \text{ssrs} - z * \text{ccrs})^2));
\]

\[
\text{PSV}\_r_z(i, j) = \text{as} * \sin(\text{thrs}) * \ldots \\
((1 - 2 * \pi^2 * f_{\text{max}}^2 * (t + 1.35 - t_{\text{peak}} + x * \text{ssrs} - z * \text{ccrs})^2) * \ldots \\
\exp(-\pi^2 * f_{\text{max}}^2 * \ldots \\
(t + 1.35 - t_{\text{peak}} + x * \text{ssrs} - z * \text{ccrs})^2));
\]

\[
\text{total}_x(i, j) = \text{SV}_i_x(i, j) + \text{SVP}\_r_x(i, j) + \text{SVSV}\_r_x(i, j) + \ldots \\
P\_i_x(i, j) + \text{PP}\_r_x(i, j) + \text{PSV}\_r_x(i, j);
\]

\[
\text{total}_z(i, j) = \text{SV}_i_z(i, j) + \text{SVP}\_r_z(i, j) + \text{SVSV}\_r_z(i, j) + \ldots \\
P\_i_z(i, j) + \text{PP}\_r_z(i, j) + \text{PSV}\_r_z(i, j);
\]

\[
i = i + 1;
\]

\[
j = j + 1;
\]

\[
\text{disp}_1(k) = \text{SV}_i_x(nz, nz) + P\_i_x(nz, nz);
\]

\[
\text{disp}_2(k) = \text{SV}_i_x(nz, nz) + P\_i_z(nz, nz);
\]

\[
\text{disp}_3(k) = \text{SVP}\_r_x(nz, nz) + \text{PP}\_r_x(nz, nz);
\]

\[
\text{disp}_4(k) = \text{SVP}\_r_x(nz, nz) + \text{PP}\_r_z(nz, nz);
\]

\[
\text{disp}_5(k) = \text{SVSV}\_r_x(nz, nz) + \text{PSV}\_r_x(nz, nz);
\]

\[
\text{disp}_6(k) = \text{SVSV}\_r_x(nz, nz) + \text{PSV}\_r_z(nz, nz);
\]

\[
\text{time_hist}(k) = t;
\]

\[
k = k + 1;
\]

\[
\% \text{Fourier transform}
\]

\[
F_s = 1 / dt;
\]

\[
NFFT = 2^\text{nextpow2}(((t_{\text{max}} - t_{\text{min}}) / dt));
\]

\[
F_1 = \text{fft}(\text{disp}_1, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
F_2 = \text{fft}(\text{disp}_2, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
F_3 = \text{fft}(\text{disp}_3, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
F_4 = \text{fft}(\text{disp}_4, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
F_5 = \text{fft}(\text{disp}_5, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
F_6 = \text{fft}(\text{disp}_6, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
F_7 = \text{fft}(\text{disp}_1 + \text{disp}_3 + \text{disp}_5, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
F_8 = \text{fft}(\text{disp}_2 + \text{disp}_4 + \text{disp}_6, NFFT) / (t_{\text{max}} - t_{\text{min}}) / dt);
\]

\[
fq = (F_s / 2) * \text{linspace}(0, 1, NFFT / 2 + 1);
\]

\[
\% \text{Rayleigh wave}
\]

\[
z = 0;
\]

\[
x = 0;
\]

\[
t = 0;
\]

\[
\omega = fq * 2 * \pi();
\]

\[
Ax = 2 * \text{abs}(F_1(1:\text{NFFT}/2+1));
\]

\[
Az = 2 * \text{abs}(F_2(1:\text{NFFT}/2+1));
\]

\[
\omega(1) = 0.000001;
\]

\[
\text{for } i = 1 : \text{max(size(fq))}
\]

\[
a = 1 * \text{abs}(\sqrt{\omega(i) / 2 \sqrt{2} - \omega(i) / 2 / Vr^2});
\]

\[
b = 1 * \text{abs}(\sqrt{\omega(i) / 2 \sqrt{2} - \omega(i) / 2 / Vs^2});
\]
for i = 1:max(size(fq))
    j = 1;
    for time = t_min:dt:t_max;
        u_x_R(i, j) = time_abs_x(i) * cos(omega(i) * (time - t_peak(i) - time_ang_x(i)));
        u_z_R(i, j) = time_abs_z(i) * cos(omega(i) * (time - t_peak(i) - time_ang_z(i)));
        j = j + 1;
    end
    i = i + 1;
end
u_x = sum(u_x_R);

% get max amplitude of R wave to scale it
max_R_x = max(u_x);

max_R_z = max(u_z);

% R wave will be recalculated on the point of interest
z = xxx;
x = zzz;

for i = 1:max(size(fq))
    a = labs(sqrt(omega(i)^2 / Vr^2 - omega(i)^2 / Vp^2));
    b = labs(sqrt(omega(i)^2 / Vr^2 - omega(i)^2 / Vs^2));
    surf_x(i) = (1i * omega(i) * Ax(i) / Vr) * (exp(a * z) - (2 * a * b * exp(b * z)) / ...)
               (b^2 + omega(i)^2 / Vr^2)) * exp(i * omega(i) * (x - Vr * t) / Vr);
    surf_z(i) = a * Az(i) * (exp(a * z) - (2 * omega(i)^2 * exp(b * z)) / Vr^2) / ...
               (b^2 + omega(i)^2 / Vr^2)) * exp(i * omega(i) * (x - Vr * t) / Vr);
end

% calculate phase angle and amplitude of R wave
for i = 1:max(size(fq))
    j = 1;
    for time = t_min:dt:t_max;
        u_x_R(i, j) = time_abs_x(i) * cos(omega(i) * (time - t_peak) - time_ang_x(i));
        u_z_R(i, j) = time_abs_z(i) * cos(omega(i) * (time - t_peak) - time_ang_z(i));
        j = j + 1;
    end
end
for i = 1:max(size(fq))
    j = 1;
    for time = t_min:dt:t_max;
        u_x_R(i, j) = time_abs_x(i) * cos(omega(i) * (time - t_peak) - time_ang_x(i));
        u_z_R(i, j) = time_abs_z(i) * cos(omega(i) * (time - t_peak) - time_ang_z(i));
        j = j + 1;
    end
end
u_x = sum(u_x_R);
u_z = sum(u_z_R);

% scale R wave
u_x = u_x * amplitude / max_R_x;
252 \( u_z = u_z \times \text{amplitude} / \text{max}_R \_x; \)
253
254 \% set amplitude of R wave as zero
255 \% \( u_x = 0; \)
256 \% \( u_z = 0; \)
257
258 \% add all displacement (R + input + reflect)
259 \( u_{xx} = u_x + \text{disp}_1 + \text{disp}_3 + \text{disp}_5; \)
260 \( u_{zz} = u_z + \text{disp}_2 + \text{disp}_4 + \text{disp}_6; \)
261
262 \% Fourier transform
263 \( F_9 = \text{fft}(u_x, \text{NFFT}) / ((t_{\text{max}} - t_{\text{min}}) / \text{dt}); \)
264 \( F_{10} = \text{fft}(u_z, \text{NFFT}) / ((t_{\text{max}} - t_{\text{min}}) / \text{dt}); \)
265 \( F_{11} = \text{fft}(u_{xx}, \text{NFFT}) / ((t_{\text{max}} - t_{\text{min}}) / \text{dt}); \)
266 \( F_{12} = \text{fft}(u_{zz}, \text{NFFT}) / ((t_{\text{max}} - t_{\text{min}}) / \text{dt}); \)
267
268 \% make final output matrix
269 \( \text{disp}_\text{out}(:, 1) = \text{time}_\text{hist}'; \)
270 \( \text{disp}_\text{out}(:, 2) = u_{xx}'; \)
271 \( \text{disp}_\text{out}(:, 4) = u_{zz}'; \)
272
273 \% figure 1: all components
274 \text{figure(1)}
275 \text{subplot(6, 2, 1)}
276 \text{plot(time}_\text{hist, disp}_1, \text{time}_\text{hist, disp}_2)
277 \text{ylim([d_min}_p l \text{ d_max}_p l])
278 \text{xlim([t_min}_p l \text{ t_max}_p l])
279 \text{xlabel('Time (s)')}
280 \text{ylabel('Displacement (m)')}
281 \text{title('Input SV + P')}
282
283 \text{subplot(6, 2, 3)}
284 \text{plot(time}_\text{hist, disp}_3, \text{time}_\text{hist, disp}_4)
285 \text{ylim([d_min}_p l \text{ d_max}_p l])
286 \text{xlim([t_min}_p l \text{ t_max}_p l])
287 \text{xlabel('Time (s)')}
288 \text{ylabel('Displacement (m)')}
289 \text{title('Reflected P')}
290
291 \text{subplot(6, 2, 5)}
292 \text{plot(time}_\text{hist, disp}_5, \text{time}_\text{hist, disp}_6)
293 \text{ylim([d_min}_p l \text{ d_max}_p l])
294 \text{xlim([t_min}_p l \text{ t_max}_p l])
295 \text{xlabel('Time (s)')}
296 \text{ylabel('Displacement (m)')}
297 \text{title('Reflected SV')}
298
299 \text{subplot(6, 2, 7)}
300 \text{plot(time}_\text{hist, disp}_1 + \text{disp}_3 + \text{disp}_5, \text{time}_\text{hist, disp}_2 + \text{disp}_4 + \text{disp}_6)
301 \text{ylim([d_min}_p l \text{ d_max}_p l])
302 \text{xlim([t_min}_p l \text{ t_max}_p l])
303 \text{xlabel('Time (s)')}
304 \text{ylabel('Displacement (m)')}
305 \text{title('Body total')}
306
307 \text{subplot(6,2,9)}
308 \text{plot(time}_\text{hist, u}_x, \text{time}_\text{hist, u}_z)
309 \text{ylim([d_min}_p l \text{ d_max}_p l])
310 \text{xlim([t_min}_p l \text{ t_max}_p l])
311 \text{xlabel('Time (s)')}
312 \text{ylabel('Displacement (m)')}
313 \text{title('Surf. Rayleigh')}
```matlab
subplot(6,2,11)
plot(time_hist, u_xx, time_hist, u_zz)
ylim([d_min_pl d_max_pl])
xlim([t_min_pl t_max_pl])
xlabel('Time (s)')
ylabel('Displacement (m)')
title('Body and Surface Total')

subplot(6,2,2)
plot(fq, 2 * abs(F_1(1:NFFT/2+1)), fq, 2 * abs(F_2(1:NFFT/2+1)))
ylim([f_amp_min_pl f_amp_max_pl])
xlim([f_min_pl f_max_pl])
xlabel('Frequency (Hz)')
ylabel('Fourier Amplitude')
title('Input SV + P')

subplot(6,2,4)
plot(fq, 2 * abs(F_3(1:NFFT/2+1)), fq, 2 * abs(F_4(1:NFFT/2+1)))
ylim([f_amp_min_pl f_amp_max_pl])
xlim([f_min_pl f_max_pl])
xlabel('Frequency (Hz)')
ylabel('Fourier Amplitude')
title('Reflected P')

subplot(6,2,6)
plot(fq, 2 * abs(F_5(1:NFFT/2+1)), fq, 2 * abs(F_6(1:NFFT/2+1)))
ylim([f_amp_min_pl f_amp_max_pl])
xlim([f_min_pl f_max_pl])
xlabel('Frequency (Hz)')
ylabel('Fourier Amplitude')
title('Reflected SV')

subplot(6,2,8)
plot(fq, 2 * abs(F_7(1:NFFT/2+1)), fq, 2 * abs(F_8(1:NFFT/2+1)))
ylim([f_amp_min_pl f_amp_max_pl])
xlim([f_min_pl f_max_pl])
xlabel('Frequency (Hz)')
ylabel('Fourier Amplitude')
title('Body Total')

subplot(6,2,10)
plot(fq, 2 * abs(F_9(1:NFFT/2+1)), fq, 2 * abs(F_10(1:NFFT/2+1)))
ylim([f_amp_min_pl f_amp_max_pl])
xlim([f_min_pl f_max_pl])
xlabel('Frequency (Hz)')
ylabel('Fourier Amplitude')
title('Surf. Rayleigh')

subplot(6,2,12)
plot(fq, 2 * abs(F_11(1:NFFT/2+1)), fq, 2 * abs(F_12(1:NFFT/2+1)))
ylim([f_amp_min_pl f_amp_max_pl])
xlim([f_min_pl f_max_pl])
xlabel('Frequency (Hz)')
ylabel('Fourier Amplitude')
title('Body and Surface Total')
```

Appendix 706

Body and Surface Wave Numerical Modeling

(In collaboration with Dr. Nima Tafazzoli, Mr. Chang-Gyun Jeong and Dr. Hexiang Wang)
706.1 Integral equations

Second method to generate wave fields is frequency-wavenumber integration method. The fk package is a frequency-wavenumber integration (Haskell, 1964; Wang and Herrmann, 1980) code developed by Zhu and Rivera (2002). The fk code package can be downloaded from http://www.eas.slu.edu/People/LZhu. In this section, using fk code package, wave propagation is simulated. The main modules of the fk code package and its function are shown as followings.

1. fk – compute Green’s functions
2. syn – compute synthetic seismogram
3. fk.pl – PERL script to simplify the use of fk

First of all, Green’s function is computed by fk with ground model properties (model layer dimension, shear and P wave velocity, density, Q and so on) and source, station (receiver) properties (source depth, epicentral distance, wave propagation direction, and receiver depth and so on). Then, Using syn with calculated Green’s function, seismograph is synthesized with given variables such as magnitude, fault strike/dip/rake, and station azimuth. Synthesized seismograph (by syn) are stored as binary / Seismic Analysis Code (SAC) form (Goldstein and Snoke, 2005). Thus, in addition to fk package, Python code to run fk package iteratively, convert results from binary to ASCII text, and make plot is developed (plot.py). All necessary variables to run fk and syn can be adjusted in plot.py. Variables are explained in next sections and can also be found in source codes (fk.f, syn,c, fk.pl, and plot.py).

706.1.1 fk3.0 package

In this section, fk3.0 package is briefly introduced. All source code can be downloaded from http://www.eas.slu.edu/People/LZhu.

706.1.1.1 fk and 'sample input'

Program fk can be run by using input text file or using PERL wrapper fk.pl as shown in next section. Example input file 'sample_input' is used to run fk. 'sample_input' is included in fk package. Total layer number, source layer number, source type, and receiver layer number are defined in the first row of input file. Layer properties are defined from second to fourth row (layer thickness / Vp / Vs / density / Qp / Qs values are defined from the first column to sixth column, respectively). Then, sigma, number of
sampling points, sampling interval, tapering factor, high / low pass filter frequencies, slowness limit, epicentral distances are defined. In example, variables can be set as following.

Line1: 3 2 2 1 0
   3 – total number of layers
   2 – source is located at the top of 2nd layer
   2 – double coupled source (0: explosion, 1: single coupled)
   1 – receiver is located at the top of 1st layer
   0 – consider both up and down going wave (1: down going wave, -1: up going wave)

Line2: 10.0000 6.3000 3.5000 2.7860 1000.00 500.00
   10.0 – depth (km)
   6.3 – Vp (km/s)
   3.5 – Vs (km/s)
   2.7860 – density (g/cm$^3$)
   1000 – Qp
   500 – Qs

Line3: 25.0000 6.3000 3.5000 2.7860 1000.00 500.00
   ref. Line2

Line4: 0.0000 8.1000 4.7000 3.3620 1600.00 800.00
   ref. Line2

Line5: 2 512 0.2 0.5 25 2 1 1
   2 – sigma in 1/trace length, the small imaginary frequency (2 – 3)
   512 – number of points in the time domain
   0.2 – time step (sec, dt)
   0.5 – tapering factor to suppress high frequencies
   25 – number of points to be saved before t0
   2 – smooth factor to increase the sampling rate
   1 1 – high pass filter (wc1, wc2)

Line6: 0. 1 0.3 15
   0. 1 – minimum and maximum slowness, to specify the window for the wavenumber integration (pmin, pmax)
0.3 – wavenumber sampling interval
15 – the maximum wavenumber at zero frequency

Line 7: 1
1 – number of distance ranges

Line 8: 200.00 20.00 200.grn.
200.00 – distance (km)
20.00 – t0
200.grn. – output file name

### 706.1.1.2 fk.pl

fk also can be run by PERL wrapper. Using the PERL wrapper fk.pl is strongly recommended by Zhu (README file of fk package). Since most of variables those are introduced on prior section are set to default values and also can be adjusted on the command line, it is much easier to use fk.pl than run fk.

### 706.1.1.3 syn

Using syn with calculated Green’s function, seismogram is computed. Direction of fault (strike / dip / rake) and recording station (azimuth), magnitude have to be defined to run syn.

### 706.1.1.4 plot.py

As mentioned above, output seismogram is in SAC form (binary). PERL script plot.py is coded to convert binary to ASCII, run fk / syn repetitively, perform Fourier transform, and plot figures. Necessary options to run fk and syn can be adjusted in plot.py.

### 706.1.2 3D seismic wave field generation using integral equation

Using fk package, 3D seismic wave field is generated. Analysis for the most common fault mechanism (strike-slip, dip-slip, and normal fault) are presented. A real earthquake example (Northridge) is also given toward the end of the section.

#### 706.1.2.1 Case 1: strike-slip fault / single layer ground

The first example is strike slip fault case. Model is as shown in Figure 706.1. Ground, fault, and wave properties are shown as below.
Figure 706.1: Ground and fault model used for analysis, results are captured on circles

- Ground properties
  - $V_S = 1\, \text{km/s}$
  - $V_P/V_S = 1.73$
  - Poisson's ratio = 0.25
  - Density = 1.32 g/cm$^3$
  - Shear modulus = 1.32 GPa
  - Elastic modulus = 3.31 GPa

- Fault properties
  - Moment magnitude = 3.5
  - Strike = 0°
  - Dip = 90°
  - Rake = 0°
  - Double - coupled source
  - Triangular source time function

- Wave properties
  - $dt = 0.1\, \text{s}$ (Max available freq. = 5 Hz, Nyquist freq.)

As indicated above, single layer ground ($V_S = 1\, \text{km/s}$) is modeled. Fault is located at 2 km depth, 2 km away from the recording points (stations). Double coupled fault source is assumed and triangular
source time function is used (Aki and Richards, 2002). Nine recording points are set as recording stations (Figure 706.1). Direction of the fault is aligned parallel to the north (strike = 0°) and recording station azimuth is set to 0°, 45°, and 90°.

Figure 706.2 – 706.10 show analyses results for the example. Legends on figures mean ‘component (epicentral distance, receiver depth)’. EW, NS, and UD components mean East - West, North - South, and Up - Down, respectively (those terms are used for all seismograms, hereafter).

Only EW components are predicted on the stations at 0° azimuth (Figure 706.2 – 706.4) and NS components are showed up on the station at 90° azimuth (Figure 706.8 – 706.10). On the station at 45° azimuth, all components (EW, NS, and UD) are observed.
Figure 706.2: Calculated time history acceleration, station azimuth = 0°
Figure 706.3: Calculated time history velocity, station azimuth = 0°
Figure 706.4: Calculated time history displacement, station azimuth = 0°.
Figure 706.5: Calculated time history acceleration, station azimuth = 45°
Figure 706.6: Calculated time history velocity, station azimuth $= 45^\circ$.
Figure 706.7: Calculated time history displacement, station azimuth $= 45^\circ$. 

- EW comp. (0.01 km, 0.0 km)
- EW comp. (0.01 km, 0.5 km)
- EW comp. (0.01 km, 1.0 km)
- EW comp. (1 km, 0.0 km)
- EW comp. (1 km, 0.5 km)
- EW comp. (1 km, 1.0 km)
- EW comp. (2 km, 0.0 km)
- EW comp. (2 km, 0.5 km)
- EW comp. (2 km, 1.0 km)
- NS comp. (0.01 km, 0.0 km)
- NS comp. (0.01 km, 0.5 km)
- NS comp. (0.01 km, 1.0 km)
- NS comp. (1 km, 0.0 km)
- NS comp. (1 km, 0.5 km)
- NS comp. (1 km, 1.0 km)
- NS comp. (2 km, 0.0 km)
- NS comp. (2 km, 0.5 km)
- NS comp. (2 km, 1.0 km)
- UD comp. (0.01 km, 0.0 km)
- UD comp. (0.01 km, 0.5 km)
- UD comp. (0.01 km, 1.0 km)
- UD comp. (1 km, 0.0 km)
- UD comp. (1 km, 0.5 km)
- UD comp. (1 km, 1.0 km)
- UD comp. (2 km, 0.0 km)
- UD comp. (2 km, 0.5 km)
- UD comp. (2 km, 1.0 km)
Figure 706.8: Calculated time history acceleration, station azimuth = 90°
Figure 706.9: Calculated time history velocity, station azimuth = 90°.
Figure 706.10: Calculated time history displacement, station azimuth = 90°
706.1.2.2 Case 2: dip-slip fault / single layer ground

Similar fault is tested for vertical (dip) slip (rake = 90°) fault case. Figure 706.11 shows model used for the analysis. Ground, fault, and wave properties are shown as below.

Figure 706.11: Ground and fault model used for analysis, results are captured on circles

- Ground properties
  - \( V_S = 1 \text{ km/s} \)
  - \( V_P/V_S = 1.73 \)
  - Poisson’s ratio = 0.25
  - Density = 1.32 g/cm\(^3\)
  - Shear modulus = 1.32 GPa
  - Elastic modulus = 3.31 GPa

- Fault properties
  - Moment magnitude = 3.5
  - Strike = 0°
  - Dip = 90°
  - Rake = 90°
  - Double - coupled source
  - Triangular source time function

- Wave properties
– \ dt = 0.1 \text{s (Max available freq. = 5 Hz, Nyquist freq.)}

Similar to the strike slip example, single layer ground (Vs = 1 km/s) is modeled. Fault is located at 2 km depth, 2 km away from the recording stations. Double coupled fault source is assumed and triangular source time function is used (Aki and Richards, 2002). Nine recording points are set as recording stations (Figure 706.11). Direction of the fault is aligned parallel to the north (strike = 0°) and rake is 90°. Station azimuth is set to 0°, 45°, and 90°.

Figure 706.12 – 706.20 show analyses results for this example. Since it’s dip slip case, permanent deformation on UD components are observed (Figure 706.17 and 706.20).
Figure 706.12: Calculated time history acceleration, station azimuth = 0°.
Figure 706.13: Calculated time history velocity, station azimuth = 0°.
Figure 706.14: Calculated time history displacement, station azimuth = $0^\circ$
Figure 706.15: Calculated time history acceleration, station azimuth = 45°
Figure 706.16: Calculated time history velocity, station azimuth = 45°
Figure 706.17: Calculated time history displacement, station azimuth = 45°
Figure 706.18: Calculated time history acceleration, station azimuth = 90°
Figure 706.19: Calculated time history velocity, station azimuth = 90°.
Figure 706.20: Calculated time history displacement, station azimuth = 90°
706.1.2.3 Case 3: normal fault / single layer ground

Normal fault is tested. Figure 706.21 shows model used for the analysis. Wave is propagated through the single layer ground \((V_s = 1 \text{ km/s})\). Properties are shown as below.

- **Ground properties**
  - \(V_s = 1 \text{ km/s}\)
  - \(V_P/V_s = 1.73\)
  - Poisson’s ratio = 0.25
  - Density = 1.32 g/cm\(^3\)
  - Shear modulus = 1.32 GPa
  - Elastic modulus = 3.31 GPa

- **Fault properties**
  - Moment magnitude = 5.0
  - Strike = 0°
  - Dip = 45°
  - Rake = 90°
  - Double - coupled source
  - Triangular source time function

- **Wave properties**
\[ dt = 0.1 \text{ s (Max available freq. = 5 Hz, Nyquist freq.)} \]

In this example, the distance between the fault and the station is increased and magnitude is changed also (Mw = 5.0). Fault is located at 30 km depth, 30 km away from the recording stations (Figure 706.21). Double coupled fault source is assumed and triangular source time function is used (Aki and Richards, 2002). Recording points are similar as prior examples (total 9 stations). Azimuth of recording station is set to 0°, 45°, and 90°.

Figure 706.22 – 706.30 show analyses results. Since the distance between fault and station is increased to 30 km and waves are propagated through the ground with relatively low shear wave velocity, arrival of propagating and reflecting waves can be observed easily (the first arrival of P wave followed by S wave). Permanent displacements by the fault movement are observed as desired at all stations (0°, 45°, and 90°).
Figure 706.22: Calculated time history acceleration, station azimuth = 0°.
Figure 706.23: Calculated time history velocity, station azimuth = 0°.
Figure 706.24: Calculated time history displacement, station azimuth = 0°
Figure 706.25: Calculated time history acceleration, station azimuth = 45°
Figure 706.26: Calculated time history velocity, station azimuth = 45°
Figure 706.27: Calculated time history displacement, station azimuth = $45^\circ$
Figure 706.28: Calculated time history acceleration, station azimuth = 90°.
Figure 706.29: Calculated time history velocity, station azimuth = 90°
Figure 706.30: Calculated time history displacement, station azimuth = 90°
706.1.2.4 Case 4: normal fault / layered ground

Normal fault within layered ground is modeled. Model is as shown Figure 706.31 and 706.32. Properties are shown below.

- Ground properties: see Figure 706.32 and Table 706.1
- Fault properties
– Moment magnitude = 5.0
– Strike = 0°
– Dip = 45°
– Rake = 90°
– Double-coupled source
– Triangular source time function

– Wave properties

– dt = 0.1 s (Max available freq. = 5 Hz, Nyquist freq.)

Fault is located at depth of 30 km, 30 km away from the recording point (station) at surface. Double coupled fault source is assumed and triangular source time function is used (Aki and Richards, 2002). Recording points are similar as prior examples (total 9 stations). The 1D standard southern California model (Hadley and Kanamori (1977), hk model hereafter) is used for ground layering. As shown in Figure 706.32, hk model is interpolated and divided to define ground layer (Table 706.1).

Results are shown on Figure 706.33 – 706.35. Since wave is propagated through the layered ground, compared to prior examples, more realistic waves are observed.
Table 706.1: Ground properties for the example

<table>
<thead>
<tr>
<th>Depth (km)</th>
<th>Thickness (km)</th>
<th>VS (km/s)</th>
<th>VP (km/s)</th>
<th>QP (km/s)</th>
<th>Poisson’s R</th>
<th>Density (g/cm³)</th>
<th>E (GPa)</th>
<th>G (GPa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.50</td>
<td>1.73</td>
<td>500</td>
<td>0.87</td>
<td>0.25</td>
<td>1.05</td>
<td>0.27</td>
</tr>
<tr>
<td>0.02</td>
<td>0.01</td>
<td>0.80</td>
<td>1.73</td>
<td>500</td>
<td>1.38</td>
<td>0.25</td>
<td>1.21</td>
<td>0.77</td>
</tr>
<tr>
<td>0.03</td>
<td>0.01</td>
<td>0.97</td>
<td>1.73</td>
<td>500</td>
<td>1.68</td>
<td>0.25</td>
<td>1.31</td>
<td>1.23</td>
</tr>
<tr>
<td>0.04</td>
<td>0.01</td>
<td>1.09</td>
<td>1.73</td>
<td>500</td>
<td>1.89</td>
<td>0.25</td>
<td>1.37</td>
<td>1.64</td>
</tr>
<tr>
<td>0.05</td>
<td>0.01</td>
<td>1.19</td>
<td>1.73</td>
<td>500</td>
<td>2.05</td>
<td>0.25</td>
<td>1.43</td>
<td>2.01</td>
</tr>
<tr>
<td>0.06</td>
<td>0.01</td>
<td>1.26</td>
<td>1.73</td>
<td>500</td>
<td>2.19</td>
<td>0.25</td>
<td>1.47</td>
<td>2.34</td>
</tr>
<tr>
<td>0.07</td>
<td>0.01</td>
<td>1.33</td>
<td>1.73</td>
<td>500</td>
<td>2.30</td>
<td>0.25</td>
<td>1.51</td>
<td>2.66</td>
</tr>
<tr>
<td>0.08</td>
<td>0.01</td>
<td>1.38</td>
<td>1.73</td>
<td>500</td>
<td>2.40</td>
<td>0.25</td>
<td>1.54</td>
<td>2.95</td>
</tr>
<tr>
<td>0.09</td>
<td>0.01</td>
<td>1.43</td>
<td>1.73</td>
<td>500</td>
<td>2.48</td>
<td>0.25</td>
<td>1.56</td>
<td>3.22</td>
</tr>
<tr>
<td>0.1</td>
<td>0.01</td>
<td>1.48</td>
<td>1.73</td>
<td>500</td>
<td>2.56</td>
<td>0.25</td>
<td>1.59</td>
<td>3.48</td>
</tr>
<tr>
<td>0.2</td>
<td>0.10</td>
<td>1.77</td>
<td>1.73</td>
<td>500</td>
<td>3.07</td>
<td>0.25</td>
<td>1.75</td>
<td>5.50</td>
</tr>
<tr>
<td>0.3</td>
<td>0.10</td>
<td>1.94</td>
<td>1.73</td>
<td>500</td>
<td>3.36</td>
<td>0.25</td>
<td>1.85</td>
<td>6.98</td>
</tr>
<tr>
<td>0.4</td>
<td>0.10</td>
<td>2.07</td>
<td>1.73</td>
<td>500</td>
<td>3.57</td>
<td>0.25</td>
<td>1.91</td>
<td>8.17</td>
</tr>
<tr>
<td>0.5</td>
<td>0.10</td>
<td>2.16</td>
<td>1.73</td>
<td>500</td>
<td>3.74</td>
<td>0.25</td>
<td>1.97</td>
<td>9.18</td>
</tr>
<tr>
<td>0.6</td>
<td>0.10</td>
<td>2.24</td>
<td>1.73</td>
<td>500</td>
<td>3.87</td>
<td>0.25</td>
<td>2.01</td>
<td>10.06</td>
</tr>
<tr>
<td>0.7</td>
<td>0.10</td>
<td>2.30</td>
<td>1.73</td>
<td>500</td>
<td>3.98</td>
<td>0.25</td>
<td>2.04</td>
<td>10.85</td>
</tr>
<tr>
<td>0.8</td>
<td>0.10</td>
<td>2.36</td>
<td>1.73</td>
<td>500</td>
<td>4.08</td>
<td>0.25</td>
<td>2.08</td>
<td>11.56</td>
</tr>
<tr>
<td>0.9</td>
<td>0.10</td>
<td>2.41</td>
<td>1.73</td>
<td>500</td>
<td>4.17</td>
<td>0.25</td>
<td>2.10</td>
<td>12.21</td>
</tr>
<tr>
<td>1.0</td>
<td>0.10</td>
<td>2.45</td>
<td>1.73</td>
<td>500</td>
<td>4.25</td>
<td>0.25</td>
<td>2.13</td>
<td>12.82</td>
</tr>
<tr>
<td>1.5</td>
<td>0.50</td>
<td>2.63</td>
<td>1.73</td>
<td>500</td>
<td>4.54</td>
<td>0.25</td>
<td>2.22</td>
<td>18.92</td>
</tr>
<tr>
<td>2.5</td>
<td>1.00</td>
<td>2.84</td>
<td>1.73</td>
<td>500</td>
<td>4.92</td>
<td>0.25</td>
<td>2.42</td>
<td>21.57</td>
</tr>
<tr>
<td>3.5</td>
<td>1.00</td>
<td>3.09</td>
<td>1.73</td>
<td>500</td>
<td>5.16</td>
<td>0.25</td>
<td>2.48</td>
<td>23.70</td>
</tr>
<tr>
<td>4.5</td>
<td>1.00</td>
<td>3.18</td>
<td>1.73</td>
<td>500</td>
<td>5.35</td>
<td>0.25</td>
<td>2.53</td>
<td>25.59</td>
</tr>
<tr>
<td>5.5</td>
<td>1.00</td>
<td>3.28</td>
<td>1.73</td>
<td>500</td>
<td>5.50</td>
<td>0.25</td>
<td>2.58</td>
<td>27.39</td>
</tr>
<tr>
<td>6.5</td>
<td>1.00</td>
<td>3.38</td>
<td>1.73</td>
<td>500</td>
<td>5.63</td>
<td>0.25</td>
<td>2.62</td>
<td>29.19</td>
</tr>
<tr>
<td>7.5</td>
<td>1.00</td>
<td>3.48</td>
<td>1.73</td>
<td>500</td>
<td>5.76</td>
<td>0.25</td>
<td>2.66</td>
<td>29.99</td>
</tr>
<tr>
<td>8.5</td>
<td>1.00</td>
<td>3.58</td>
<td>1.73</td>
<td>500</td>
<td>5.89</td>
<td>0.25</td>
<td>2.70</td>
<td>30.79</td>
</tr>
<tr>
<td>9.5</td>
<td>1.00</td>
<td>3.68</td>
<td>1.73</td>
<td>500</td>
<td>6.02</td>
<td>0.25</td>
<td>2.74</td>
<td>31.59</td>
</tr>
<tr>
<td>10.0</td>
<td>1.00</td>
<td>3.78</td>
<td>1.73</td>
<td>500</td>
<td>6.15</td>
<td>0.25</td>
<td>2.78</td>
<td>32.39</td>
</tr>
</tbody>
</table>
Figure 706.33: Calculated time history acceleration, station azimuth = 0°.
Figure 706.34: Calculated time history velocity, station azimuth = 0°.
Figure 706.35: Calculated time history displacement, station azimuth = 0°.
Figure 706.36: Calculated time history acceleration, station azimuth = 45°
Figure 706.37: Calculated time history velocity, station azimuth = 45°
Figure 706.38: Calculated time history displacement, station azimuth = 45°
Figure 706.39: Calculated time history acceleration, station azimuth = 90°
Figure 706.40: Calculated time history velocity, station azimuth = 90°
Figure 706.41: Calculated time history displacement, station azimuth = 90°
706.1.2.5 Case 5: Northridge earthquake / layered ground

In this example, Northridge earthquakes are simulated. Northridge earthquake has occurred on January, 1994 with a moment magnitude of 6.7. Properties are shown as below and partly adapted from Hisada (2008) and Wald et al. (1996).

- Ground properties: see Table 706.2

- Fault properties
  - Moment magnitude = 6.7
  - Strike = 122°
  - Dip = 40°
  - Rake = 140°
  - Double-coupled source
  - Triangular source time function

- Wave properties
  - $dt = 0.05 \text{ s} \ (\text{Max available freq. } = 10 \text{ Hz, Nyquist freq.})$

Source depth is set as 25 km and epicentral distance from the fault to the station is set as 30 km. Station is located on the ground surface. As shown in Table 706.2, ground is divided into 9 layers (Hisada, 2008).

Figure 706.42 shows analyses results. Computed results are compared with measured one. Measured records are obtained from cosmos virtual data center http://db.cosmos-eq.org/. As shown in Figure 706.42, predicted seismogram agrees well with measured ones considering the fk package assumes a single point source and simplified ground.pdf.
Table 706.2: Ground properties for the example

<table>
<thead>
<tr>
<th>Depth (km)</th>
<th>Thickness (km)</th>
<th>VS (km/s)</th>
<th>VP/VS</th>
<th>QB (km/s)</th>
<th>VP (km/s)</th>
<th>Poisson’s R</th>
<th>Density (g/cm³)</th>
<th>G (GPa)</th>
<th>E (GPa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.05</td>
<td>0.30</td>
<td>1.730</td>
<td>600</td>
<td>0.52</td>
<td>0.25</td>
<td>0.94</td>
<td>0.08</td>
<td>0.21</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>0.40</td>
<td>1.730</td>
<td>600</td>
<td>0.69</td>
<td>0.25</td>
<td>0.99</td>
<td>0.16</td>
<td>0.40</td>
</tr>
<tr>
<td>0.2</td>
<td>0.10</td>
<td>0.50</td>
<td>1.730</td>
<td>600</td>
<td>0.87</td>
<td>0.25</td>
<td>1.05</td>
<td>0.26</td>
<td>0.65</td>
</tr>
<tr>
<td>0.3</td>
<td>0.10</td>
<td>0.75</td>
<td>1.730</td>
<td>600</td>
<td>1.30</td>
<td>0.25</td>
<td>1.19</td>
<td>0.67</td>
<td>1.67</td>
</tr>
<tr>
<td>0.5</td>
<td>0.20</td>
<td>1.00</td>
<td>1.730</td>
<td>600</td>
<td>1.73</td>
<td>0.25</td>
<td>1.32</td>
<td>1.32</td>
<td>3.31</td>
</tr>
<tr>
<td>1.5</td>
<td>1.00</td>
<td>2.00</td>
<td>1.730</td>
<td>600</td>
<td>3.46</td>
<td>0.25</td>
<td>1.88</td>
<td>7.51</td>
<td>18.76</td>
</tr>
<tr>
<td>4</td>
<td>2.50</td>
<td>3.20</td>
<td>1.730</td>
<td>600</td>
<td>5.54</td>
<td>0.25</td>
<td>2.54</td>
<td>26.03</td>
<td>65.02</td>
</tr>
<tr>
<td>27</td>
<td>23.00</td>
<td>3.60</td>
<td>1.730</td>
<td>600</td>
<td>6.23</td>
<td>0.25</td>
<td>2.76</td>
<td>35.81</td>
<td>89.46</td>
</tr>
<tr>
<td>40</td>
<td>13.00</td>
<td>3.90</td>
<td>1.730</td>
<td>900</td>
<td>6.75</td>
<td>0.25</td>
<td>2.93</td>
<td>44.55</td>
<td>111.30</td>
</tr>
</tbody>
</table>
Figure 706.42: Analysis result, thick line is observed (SYL station, http://db.cosmos-eq.org/); results presented by the thin line are calculated using fk/syn
Appendix 707

Real-ESSI Illustrative Examples


(In collaboration with Dr. Yuan Feng, Prof. José Abell, Mr. Sumeet Kumar Sinha, Dr. Han Yang and Dr. Hexiang Wang)
This chapter presents a number of illustrative examples. The main aim is simple: present Real-ESSI Simulator features (available elements, algorithms, domain specific language (DSL), &c.) through a number of simple examples. It is noted that all presented elements and algorithms work in sequential and parallel mode. However, presented examples are very small, and parallel mode will not bring any benefits.
707.1 Elastic Beam Element Under Static Loading

This is a simple beam example under static loading in three directions. The diagram below shows the loading in one bending direction.

![Diagram of a cantilever beam with a downward force applied at the right end.]

Figure 707.1: The cantilever model

**ESSI model fei/DSL file:**

```
model name "beam_1element" ;
// define the node coordinates
add node # 1 at ( 0.0*m , 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at ( 1.0*m , 0.0*m, 0.0*m) with 6 dofs;
// Geometry: width and height. Help the beam definition.
b=0.2*m;
h=0.2*m;
I=b*h^3/12.0;
// define the beam element
add element # 1 type beam_elastic with nodes (1,2)
cross_section = b*h
elastic_modulus = 1e9*N/m^2
shear_modulus = 5e8*N/m^2
torsion_Jx = 0.33*b*h^3
bending_Iy = I
bending_Iz = I
mass_density = 0*kg/m^3
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
// add boundary condition
fix node # 1 dofs all;
// axial loading
new loading stage "axial";
add load # 1 to node # 2 type linear Fx = 1*N;
define load factor increment 1;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 1 steps using static algorithm;
```
// bending in one direction
new loading stage "bending1";
remove load # 1;
add load # 2 to node # 2 type linear Fy = 1*N;
define load factor increment 1;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 1 steps using static algorithm;
// bending in the other direction
new loading stage "bending2";
remove load # 2;
add load # 3 to node # 2 type linear Fz = 1*N;
define load factor increment 1;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 1 steps using static algorithm;
bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.2 Elastic Beam Element under Dynamic Loading

Problem description:

![Beam model diagram](image)

Figure 707.2: The cantilever model.

ESSI model fei/DSL file:

```plaintext
model name "beam_1element";

// add node
add node #1 at (0.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node #2 at (1.0*m, 0.0*m, 0.0*m) with 6 dofs;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I = b*h^3/12.0;
A = b*h;
L = 1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
posion_ratio = 0.3;

// add elements
add element #1 type beam_elastic with nodes (1,2)
cross_section = b*h
elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2/(1+posion_ratio)
torsion_Jx = 0.33*b*h^3
bending_Iy = b*h^3/12
bending_Iz = b*h^3/12
mass_density = rho
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
```
joint_2_offset = (0*m, 0*m, 0*m);

// add boundary condition
fix node # 1 dofs all;

// // --slowLoading---------------------------------------------------------------
// add load in 180 seconds. (Slow)
new loading stage "slowLoading";
add load # 1 to node # 2 type path_time_series
Fz = 1.*N
series_file = "slowLoading.txt";
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 2000 steps using transient algorithm
time_step = 0.1*s;

// // --fastLoading---------------------------------------------------------------
// add load in 0.6 seconds (Fast)
remove load # 1;
new loading stage "fastLoading";
add load # 2 to node # 2 type path_time_series
Fz = 1.*N
series_file = "fastLoading.txt";
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 1000 steps using transient algorithm
time_step = 0.01*s;

// // --freeVibration-------------------------------------------------------------
// add a load and then release to free vibration
remove load # 2;
new loading stage "freeVibration";
add load # 3 to node # 2 type path_time_series
Fz = 1.*N
series_file = "freeVibration.txt";
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 2000 steps using transient algorithm
time_step = 0.01*s;

bye;
Displacement Results

Figure 707.3: Slow loading condition, vertical displacements or the cantilever tip.

Figure 707.4: Fast loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.5: Free vibration, vertical displacements of the cantilever tip.
707.3 Cantilever, 5 Elastic Beam Elements

Problem description:

![Cantilever Model](image)

Figure 707.6: The cantilever model.

ESSI model fei/DSL file:

```plaintext
model name "beam_5element";

// add node
add node # 1 at (0.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at (0.2*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 3 at (0.4*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 4 at (0.6*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 5 at (0.8*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 6 at (1.0*m, 0.0*m, 0.0*m) with 6 dofs;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
possion_ratio=0.3;

// Cross section geometry: width and height
b=0.2*m;
h=0.2*m;

// add elements
ii=1;
while (ii<6) {

```
add element # ii type beam_elastic with nodes (ii,ii+1)
  cross_section = b*h
  elastic_modulus = elastic_constant
  shear_modulus = elastic_constant/2/(1+possion_ratio)
  torsion_Jx = 0.33*b*h^3/12
  bending_Iy = b*h^3/12
  bending_Iz = b*h^3/12
  mass_density = rho
  xz_plane_vector = ( 1, 0, 1)
  joint_1_offset = (0*m, 0*m, 0*m)
  joint_2_offset = (0*m, 0*m, 0*m);
  ii+=1;
}

// add boundary condition
fix node # 1 dofs all;

// // ---------------------------------------------------------------------------
// // --slowLoading---------------------------------------------------------------
// // add load in 180 seconds.
// // // ---------------------------------------------------------------------------
// new loading stage "slowLoading";
// add load # 1 to node # 6 type path_time_series
  // Fz = 1.*N
  // series_file = "slowLoading.txt" ;
  // define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
  // define algorithm With_no_convergence_check ;
  // define solver ProfileSPD;
  // simulate 2000 steps using transient algorithm
  // time_step = 0.1*s;

// // // ---------------------------------------------------------------------------
// // --fastLoading---------------------------------------------------------------
// // add load in 0.6 seconds.
// // // ---------------------------------------------------------------------------
// remove load # 1;
// new loading stage "fastLoading";
// add load # 2 to node # 6 type path_time_series
  // Fz = 1.*N
  // series_file = "fastLoading.txt" ;
  // define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
  // define algorithm With_no_convergence_check ;
  // define solver ProfileSPD;
  // simulate 1000 steps using transient algorithm
  // time_step = 0.01*s;

// // // ---------------------------------------------------------------------------
// // --freeVibration-------------------------------------------------------------
// // add a load and then release for free vibration
// // // ---------------------------------------------------------------------------
// remove load # 2;
new loading stage "freeVibration";
add load # 3 to node # 6 type path_time_series
Fz = 1.*N
series_file = "freeVibration.txt" ;
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.1*s;
bye;

Displacement results

Figure 707.7: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.8: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.9: Free vibration condition, vertical displacements of the cantilever tip.
707.4 Cantilever, One 27 Node Brick Element, Dynamic Loading

Problem description:

![Cantilever model](image)

Figure 707.10: The cantilever model.

**ESSI model fei/DSL file:**

```plaintext
model name "brick_1element" ;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
poisson_ratio=0.3;

add material # 1 type linear_elastic_isotropic_3d_LT
mass_density = rho
elastic_modulus = elastic_constant
poisson_ratio = poisson_ratio;

add node # 1 at (0.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 2 at (0.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 3 at (1.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 4 at (1.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 5 at (0.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 6 at (1.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 7 at (1.0000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 8 at (0.0000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 9 at (0.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
```

add node # 10 at (0.5000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 11 at (1.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
add node # 12 at (0.5000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 13 at (0.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add node # 14 at (0.5000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 15 at (1.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add node # 16 at (0.5000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 17 at (0.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 18 at (0.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 19 at (1.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 20 at (1.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 21 at (0.5000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 22 at (0.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 23 at (0.5000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 24 at (1.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 25 at (0.5000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 26 at (0.5000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
add node # 27 at (0.5000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add element # 1 type 27NodeBrickLT with nodes(2, 1, 3, 4, 5, 8, 7, 6, 9, 10, ←
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27) use ←
material # 1;
fix node # 1 dofs all;
fix node # 2 dofs all;
fix node # 5 dofs all;
fix node # 8 dofs all;
fix node # 9 dofs all;
fix node # 13 dofs all;
fix node # 17 dofs all;
fix node # 18 dofs all;
fix node # 22 dofs all;

// // --slowLoading---------------------------------------------------------------
// new loading stage "slowLoading";
// add load # 1 to node # 4 type path_time_series Fz=1/36.0*N series_file = ←
"slowLoading.txt" ;
// add load # 2 to node # 6 type path_time_series Fz=1/36.0*N series_file = ←
"slowLoading.txt" ;
// add load # 3 to node # 3 type path_time_series Fz=1/36.0*N series_file = ←
"slowLoading.txt" ;
// add load # 4 to node # 7 type path_time_series Fz=1/36.0*N series_file = ←
"slowLoading.txt" ;
// add load # 5 to node # 20 type path_time_series Fz=1/9.0*N series_file = ←
"slowLoading.txt" ;
// add load # 6 to node # 11 type path_time_series Fz=1/9.0*N series_file = ←
"slowLoading.txt" ;
// add load # 7 to node # 15 type path_time_series Fz=1/9.0*N series_file = ←
/* slowLoading.txt */
75 // add load # 8 to node # 19 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
76 // add load # 9 to node # 24 type path_time_series Fz=4/9.0*N series_file = "slowLoading.txt";
77 // add algorithm and solver
78 // define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
79 // define algorithm With_no_convergence_check;
80 // define solver ProfileSPD;
81 // simulate 2000 steps using transient algorithm
82 // time_step = 0.1*s;
83
84 // ---fastLoading---------------------------------------------------------------
85 // new loading stage "fastLoading";
86 // add load # 101 to node # 4 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
87 // add load # 102 to node # 6 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
88 // add load # 103 to node # 3 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
89 // add load # 104 to node # 7 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
90 // add load # 105 to node # 20 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
91 // add load # 106 to node # 11 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
92 // add load # 107 to node # 15 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
93 // add load # 108 to node # 19 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
94 // add load # 109 to node # 24 type path_time_series Fz=4/9.0*N series_file = "fastLoading.txt";
95 // add algorithm and solver
96 // define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
97 // define algorithm With_no_convergence_check;
98 // define solver ProfileSPD;
99 // simulate 1000 steps using transient algorithm
100 // time_step = 0.01*s;
101
102 // ---freeVibration------------------------------------------------------------
103 // new loading stage "freeVibration";
104 add load # 201 to node # 4 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
105 add load # 202 to node # 6 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
106 add load # 203 to node # 3 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";}
add load # 204 to node # 7 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
add load # 205 to node # 20 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
add load # 206 to node # 11 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
add load # 207 to node # 15 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
add load # 208 to node # 19 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
add load # 209 to node # 24 type path_time_series Fz=4/9.0*N series_file = "freeVibration.txt";

// add algorithm and solver
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 10000 steps using transient algorithm
time_step = 0.001*s;

// end
bye;

Displacement results against time series

Figure 707.11: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.12: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.13: Free vibration condition, vertical displacements of the cantilever tip.
707.5 Simulate Cantilever Using Five 27 Node Brick Elements

Problem description:

![Cantilever model diagram](image)

Figure 707.14: The cantilever model.

ESSI model fei/DSL file:

```plaintext
model name "brick_5element";

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
possion_ratio=0.3;

add material # 1 type linear_elastic_isotropic_3d_LT
mass_density = rho
elastic_modulus = elastic_constant
poisson_ratio = possion_ratio;

add node # 1 at (0.0*m, 0.0*m, 0.0*m) with 3 dofs;
add node # 2 at (0.1*m, 0.0*m, 0.0*m) with 3 dofs;
add node # 3 at (0.2*m, 0.0*m, 0.0*m) with 3 dofs;
add node # 4 at (0.0*m, 0.1*m, 0.0*m) with 3 dofs;
add node # 5 at (0.1*m, 0.1*m, 0.0*m) with 3 dofs;
add node # 6 at (0.2*m, 0.1*m, 0.0*m) with 3 dofs;
add node # 7 at (0.0*m, 0.2*m, 0.0*m) with 3 dofs;
add node # 8 at (0.1*m, 0.2*m, 0.0*m) with 3 dofs;
add node # 9 at (0.2*m, 0.2*m, 0.0*m) with 3 dofs;
```

fix node No 1 dofs ux uy uz;
fix node No 2 dofs ux uy uz;
fix node No 3 dofs ux uy uz;
fix node No 4 dofs ux uy uz;
fix node No 5 dofs ux uy uz;
fix node No 6 dofs ux uy uz;
fix node No 7 dofs ux uy uz;
fix node No 8 dofs ux uy uz;
fix node No 9 dofs ux uy uz;
e = 0;
hh = 0*m;
NBricks=5;
dz = 0.2*m;
while ( e < NBricks)
{
    hh += dz;
    add node # 10+18*e at (0.0*m, 0.0*m , hh - 0.5*dz) with 3 dofs;
    add node # 11+18*e at (0.1*m, 0.0*m , hh - 0.5*dz) with 3 dofs;
    add node # 12+18*e at (0.2*m, 0.0*m , hh - 0.5*dz) with 3 dofs;
    add node # 13+18*e at (0.0*m, 0.1*m , hh - 0.5*dz) with 3 dofs;
    add node # 14+18*e at (0.1*m, 0.1*m , hh - 0.5*dz) with 3 dofs;
    add node # 15+18*e at (0.2*m, 0.1*m , hh - 0.5*dz) with 3 dofs;
    add node # 16+18*e at (0.0*m, 0.2*m , hh - 0.5*dz) with 3 dofs;
    add node # 17+18*e at (0.1*m, 0.2*m , hh - 0.5*dz) with 3 dofs;
    add node # 18+18*e at (0.2*m, 0.2*m , hh - 0.5*dz) with 3 dofs;
    add node # 19+18*e at (0.0*m, 0.0*m , hh) with 3 dofs;
    add node # 20+18*e at (0.1*m, 0.0*m , hh) with 3 dofs;
    add node # 21+18*e at (0.2*m, 0.0*m , hh) with 3 dofs;
    add node # 22+18*e at (0.0*m, 0.1*m , hh) with 3 dofs;
    add node # 23+18*e at (0.1*m, 0.1*m , hh) with 3 dofs;
    add node # 24+18*e at (0.2*m, 0.1*m , hh) with 3 dofs;
    add node # 25+18*e at (0.0*m, 0.2*m , hh) with 3 dofs;
    add node # 26+18*e at (0.1*m, 0.2*m , hh) with 3 dofs;
    add node # 27+18*e at (0.2*m, 0.2*m , hh) with 3 dofs;
    add element # e+1 type 27NodeBrickLT with nodes
        (21+18*e,
         27+18*e,
         25+18*e,
         19+18*e,
         3+18*e,
         9+18*e,
         7+18*e,
         1+18*e,
         24+18*e,
         26+18*e,
22+18*e,
20+18*e,
6+18*e,
8+18*e,
4+18*e,
2+18*e,
12+18*e,
18+18*e,
16+18*e,
10+18*e,
14+18*e,
15+18*e,
17+18*e,
13+18*e,
11+18*e,
23+18*e,
5+18*e
)
use material # 1;
e += 1;
};
e = e -1;

// // --slowLoading---------------------------------------------------------------
// add the 1 Newton load in 180 seconds.
// // --slowLoading---------------------------------------------------------------
// new loading stage "slowLoading";
// add load # 1 to node # (19+18*e) type path_time_series Fx=1/36.0*N ←
series_file = "slowLoading.txt";
// add load # 2 to node # (20+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";
// add load # 3 to node # (21+18*e) type path_time_series Fx=1/36.0*N ←
series_file = "slowLoading.txt";
// add load # 4 to node # (22+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";
// add load # 5 to node # (23+18*e) type path_time_series Fx=4/9.0*N ←
series_file = "slowLoading.txt";
// add load # 6 to node # (24+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";
// add load # 7 to node # (25+18*e) type path_time_series Fx=1/36.0*N ←
series_file = "slowLoading.txt";
// add load # 8 to node # (26+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";
// add load # 9 to node # (27+18*e) type path_time_series Fx=1/36.0*N ←
series_file = "slowLoading.txt";
// add algorithm and solver
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check;
// define solver ProfileSPD;
// simulate 2000 steps using transient algorithm
// time_step = 0.1*s;

// // --fastLoading---------------------------------------------------------------
// add the 1 Newton load in 0.6 seconds.
// // --freeVibration---------------------------------------------------------------
// new loading stage "freeVibration";
// add a load and then release to free vibration
add load # 204 to node # (22+18*e) type path_time_series Fx=1/9.0*N series_file ← "freeVibration.txt" ;
add load # 205 to node # (23+18*e) type path_time_series Fx=4/9.0*N series_file ← "freeVibration.txt" ;
add load # 206 to node # (24+18*e) type path_time_series Fx=1/9.0*N series_file ← "freeVibration.txt" ;
add load # 207 to node # (25+18*e) type path_time_series Fx=1/36.0*N ← series_file = "freeVibration.txt" ;
add load # 208 to node # (26+18*e) type path_time_series Fx=1/9.0*N series_file ← "freeVibration.txt" ;
add load # 209 to node # (27+18*e) type path_time_series Fx=1/36.0*N ← series_file = "freeVibration.txt" ;
// add algorithm and solver
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.1*s;
// end
bye;

Displacement Results.

Figure 707.15: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.16: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.17: Free vibration condition, vertical displacements of the cantilever tip.
707.6 Elastic Beam Element under Dynamic Loading with concentrated mass

Problem description:

Figure 707.18: The cantilever-mass model.

ESSI model fei/DSL file:

```plaintext
model name "beam-mass_1element" ;

// add node
add node # 1 at ( 0.0*m , 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at ( 1.0*m , 0.0*m, 0.0*m) with 6 dofs;

// Geometry: width and height
b=0.2*m;

h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;

A=b*h;
L=1*m;

rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);

poisson_ratio=0.3;

// add elements
add element # 1 type beam_elastic with nodes (1,2)
cross_section = b*h
elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2/(1+poisson_ratio)
torsion_Jx = 0.33*b*h^3
```

Figure 707.18: The cantilever-mass model.
bending_Iy = b*h^3/12
bending_Iz = b*h^3/12
mass_density = rho
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

// add boundary condition
fix node # 1 dofs all;

// add mass
beamMass=rho*A*L;
add mass to node # 2
mx = beamMass
my = beamMass
mz = beamMass
Imx = 0*beamMass*L^2
Imy = 0*beamMass*L^2
Imz = 0*beamMass*L^2;

// // --slowLoading---------------------------------------------------------------
// new loading stage "slowLoading";
// add load # 1 to node # 2 type path_time_series
// Fz = 1.*N
// series_file = "slowLoading.txt" ;
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check ;
// define solver ProfileSPD;
// simulate 2000 steps using transient algorithm
// time_step = 0.1*s;

// // --fastLoading---------------------------------------------------------------
// remove load # 1;
// new loading stage "fastLoading";
// add load # 2 to node # 2 type path_time_series
// Fz = 1.*N
// series_file = "fastLoading.txt" ;
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check ;
// define solver ProfileSPD;
// simulate 1000 steps using transient algorithm
// time_step = 0.01*s;

// // --freeVibration-------------------------------------------------------------
// remove load # 2;
new loading stage "freeVibration";
add load # 3 to node # 2 type path_time_series
  Fz = 1.*N
series_file = "freeVibration.txt" ;
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 1000 steps using transient algorithm
time_step = 0.01*s;
bye;

Displacement results against time series

Figure 707.19: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.20: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.21: Free vibration condition, vertical displacements of the cantilever tip.
707.7 Elastic Beam, 27 Node Brick Model With Concentrated Mass

Problem description:

![Diagram of the cantilever-mass model.](image)

Figure 707.22: The cantilever-mass model.

**ESSI model fei/DSL file:**

```plaintext
model name "brick-mass_1element" ;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
possion_ratio=0.3;

add material # 1 type linear_elastic_isotropic_3d_LT
  mass_density = rho
  elastic_modulus = elastic_constant
  poisson_ratio = possion_ratio;

add node # 1 at ( 0.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 2 at ( 0.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 3 at ( 1.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 4 at ( 1.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 5 at ( 0.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 6 at ( 1.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 7 at ( 1.0000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 8 at ( 0.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 9 at ( 0.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
add node # 10 at ( 0.5000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
```

Jeremić et al., University of California, Davis version: 28. May, 2021, 17:09
add node # 11 at (1.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
add node # 12 at (0.5000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 13 at (0.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add node # 14 at (0.5000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 15 at (1.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add node # 16 at (0.5000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 17 at (0.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 18 at (0.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 19 at (1.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 20 at (1.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 21 at (0.5000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 22 at (0.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 23 at (0.5000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 24 at (1.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 25 at (0.5000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 26 at (0.5000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
add node # 27 at (0.5000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add element # 1 type 27NodeBrickLT with nodes(2, 1, 3, 4, 5, 8, 7, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27) use material # 1;
fix node # 1 dofs all;
fix node # 2 dofs all;
fix node # 5 dofs all;
fix node # 8 dofs all;
fix node # 9 dofs all;
fix node # 13 dofs all;
fix node # 17 dofs all;
fix node # 18 dofs all;
fix node # 22 dofs all;
// Mapping from 3 dofs to 6 dofs.
add node # 1003 at (1.0000 *m, 0.2000 *m, 0.0000 *m) with 6 dofs;
add node # 1004 at (1.0000 *m, 0.0000 *m, 0.0000 *m) with 6 dofs;
add node # 1006 at (1.0000 *m, 0.0000 *m, 0.2000 *m) with 6 dofs;
add node # 1007 at (1.0000 *m, 0.2000 *m, 0.2000 *m) with 6 dofs;
// And connect the nodes at the same location.
add constraint equal dof with master node # 3 and slave node # 1003 dof to constrain ux uy uz;
add constraint equal dof with master node # 4 and slave node # 1004 dof to constrain ux uy uz;
add constraint equal dof with master node # 6 and slave node # 1006 dof to constrain ux uy uz;
add constraint equal dof with master node # 7 and slave node # 1007 dof to constrain ux uy uz;
add mass to node # 24 mx = rho*A*L my = rho*A*L mz = rho*A*L;
// add 6 beams to connect the mass
```plaintext
smallb = 0.01*m;
smallh = 0.01*m;
smallE = 1e9*N/m^2;
smallnu = 0.3;
smallrhom = 0*kg/m^3;
smallI = smallb*smallh^3/12.0;

add element # 11 type beam_elastic with nodes (1003,1004)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrhom
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

add element # 12 type beam_elastic with nodes (1003,1006)
cross_section = smallb*smallh
elastic_modulus = smallE
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrhom
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

add element # 13 type beam_elastic with nodes (1003,1007)
cross_section = smallb*smallh
elastic_modulus = smallE
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrhom
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

add element # 14 type beam_elastic with nodes (1004,1006)
cross_section = smallb*smallh
elastic_modulus = smallE
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrhom
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

add element # 15 type beam_elastic with nodes (1004,1007)
cross_section = smallb*smallh
elastic_modulus = smallE
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrhom
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
```

cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

add element # 16 type beam_elastic with nodes (1006,1007)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

// // --slowLoading---------------------------------------------------------------
// // add the 1 Newton load in 180 seconds.
// add algorithm and solver
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check ;
// define solver ProfileSPD;
// simulate 2000 steps using transient algorithm
170 // time_step = 0.1*s;
171 // // --fastLoading---------------------------------------------------------------
172 // add the 1 Newton load in 0.6 seconds.
173 // // --fastLoading---------------------------------------------------------------
174 // new loading stage "fastLoading";
175 // add load # 101 to node # 4 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
176 // add load # 102 to node # 6 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
177 // add load # 103 to node # 3 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
178 // add load # 104 to node # 7 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
179 // add load # 105 to node # 20 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
180 // add load # 106 to node # 11 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
181 // add load # 107 to node # 15 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
182 // add load # 108 to node # 19 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
183 // add load # 109 to node # 24 type path_time_series Fz=4/9.0*N series_file = "fastLoading.txt";
184 // add algorithm and solver
185 // define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
186 // define algorithm With_no_convergence_check ;
187 // define solver ProfileSPD;
188 // simulate 1000 steps using transient algorithm
189 // time_step = 0.01*s;
190 // // --freeVibration---------------------------------------------------------------
191 // new loading stage "freeVibration";
192 // add load # 201 to node # 4 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
193 // add load # 202 to node # 6 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
194 // add load # 203 to node # 3 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
195 // add load # 204 to node # 7 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
196 // add load # 205 to node # 20 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
197 // add load # 206 to node # 11 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
198 // add load # 207 to node # 15 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
```plaintext
add load # 208 to node # 19 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
add load # 209 to node # 24 type path_time_series Fz=4/9.0*N series_file = "freeVibration.txt";
// add algorithm and solver
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.1*s;

// end
bye;
```

**Displacement Results.**

![Displacement Graph](image)

Figure 707.23: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded [here](#).
Figure 707.24: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.25: Free vibration condition, vertical displacements of the cantilever tip.
707.8 Elastic Beam Element, Dynamic Loading, Viscous (Rayleigh/Caughey) and Numerical (Newmark/HHT) Damping

Problem description:

![Diagram of cantilever-mass model](image)

Figure 707.26: The cantilever-mass model.

**ESSI model fei/DSL file:**

```plaintext
model name "beam_1element" ;

// add node
add node # 1 at ( 0.0*m , 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at ( 1.0*m , 0.0*m, 0.0*m) with 6 dofs;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
possion_ratio=0.3;

// add elements
add element # 1 type beam_elastic with nodes (1,2)
cross_section = b*h
elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2/(1+possion_ratio)
torsion_Jx = 0.33*b*h^3
bending_Iy = b*h^3/12
bending_Iz = b*h^3/12
mass_density = rho
```

Jeremić et al., University of California, Davis
version: 28 May, 2021, 17:09
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

// add boundary condition
fix node # 1 dofs all;

// // --no-damping-------------------------------------------------------------
// // --hht-damping-------------------------------------------------------------
// remove load # 3;
// new loading stage "HHT-damping";
// add load # 4 to node # 6 type path_time_series
// Fz = 1.*kN
// series_file = "freeVibration.txt"
// define dynamic integrator Hilber_Hughes_Taylor with alpha = -0.20;
// define algorithm With_no_convergence_check
// define solver ProfileSPD;
// simulate 300 steps using transient algorithm
// time_step = 0.1*s;

// // --rayleigh-damping--------------------------------------------------------
// // --newmark-damping----------------------------------------------------------
// remove load # 3;
// new loading stage "HHT-damping";
// add load # 4 to node # 6 type path_time_series
// Fz = 1.*kN
// series_file = "freeVibration.txt"
// define dynamic integrator Hilber_Hughes_Taylor with alpha = -0.20;
// define algorithm With_no_convergence_check
// define solver ProfileSPD;
// simulate 300 steps using transient algorithm
// time_step = 0.1*s;

// // --newmark-damping----------------------------------------------------------
// // --no-damping---------------------------------------------------------------
// // new loading stage "no-damping";
// add load # 1 to node # 2 type path_time_series
// Fz = 1.*N
// series_file = "freeVibration.txt";
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check
// define solver ProfileSPD;
// simulate 100 steps using transient algorithm
// time_step = 0.1*s;
Jeremić et al., Real-ESSI

Real-ESSI Lecture Notes

79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129

707.8. ELASTIC BEAM ELEMENT, DYNAMIC . . .

page: 2846 of 3091

// remove load # 4;
// simulate using eigen algorithm number_of_modes = 2;
f1=0.996807/s;
f2=0.996807/s;
w1 = 2*pi*f1;
w2 = 2*pi*f2;
xi=0.05;
rayl_a1 = 2*xi/(w1 + w2);
rayl_a0 = rayl_a1*w1*w2;
add damping # 1 type Rayleigh with
a0 = rayl_a0
a1 = rayl_a1
stiffness_to_use = Initial_Stiffness;
add damping # 1 to element # 1;
new loading stage "Rayleigh-damping";
add load # 5 to node # 2 type path_time_series
Fz = 1.*N
series_file = "freeVibration.txt" ;
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.1*s;
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//

// ---------------------------------------------------------------------------// --Caughey3rd-damping-------------------------------------------------------// ---------------------------------------------------------------------------add damping # 2 type Caughey3rd with
a0 = 0.560523/s
a1 = 0.0730746*s
a2 = 0.000361559*s^3
stiffness_to_use = Last_Committed_Stiffness;
kk=1;
while (kk<6) {
add damping # 2 to element # kk;
kk+=1;
}
new loading stage "Caughey3rd-damping";
add load # 6 to node # 6 type path_time_series
Fz = 10.*kN
series_file = "freeVibration.txt" ;
For Caughey3rd damping, we have to add some Newmark damping,
Otherwise, there will be some high frequency noise.
define dynamic integrator Newmark with gamma = 0.6 beta = 0.3025;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.2*s;

Jeremić et al.

University of California, Davis



Displacement results against time series

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.27: Free vibration condition, no damping, vertical displacements of the cantilever tip.

Figure 707.28: Free vibration condition, viscous (Rayleigh) damping, vertical displacements of the cantilever tip.
Figure 707.29: Free vibration condition, viscous (Caughey3rd) damping, vertical displacements of the cantilever tip.

Figure 707.30: Free vibration condition, viscous (Caughey4th) damping, vertical displacements of the cantilever tip.
Figure 707.31: Free vibration condition, numerical (Newmark) damping, vertical displacements of the cantilever tip.

Figure 707.32: Free vibration condition, numerical (HHT) damping, vertical displacements of the cantilever tip.
### 707.9 Elastic Beam Element for a Simple Frame Structure

**Problem Description**

- **Dimensions:** width = 6 m, height = 6 m, force = 100 N
- **Element dimensions:** length = 6 m, cross section width = 1 m, cross section height = 1 m, mass density $\rho = 0.0 \text{kN/m}^3$, Young’s modulus $E = 1E8 \text{ Pa}$, Poisson’s ratio $\nu = 0.0$.

![Elastic frame with beam elastic elements](image)

**ESSI model fei/DSL file:**

```plaintext
model name "beam_element_presentation";
add node # 1 at ( 0.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 2 at ( 0.00*m, 0.00*m, 6.00*m) with 6 dofs;
add node # 3 at ( 6.00*m, 0.00*m, 6.00*m) with 6 dofs;
add node # 4 at ( 6.00*m, 0.00*m, 0.00*m) with 6 dofs;
elastic_constant = 1e8*N/m^2;
b=1*m;
h=1*m;
rho = 0*kg/m^3; // Mass density
add element # 1 type beam_elastic with nodes (1, 2)
cross_section = b*h elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2
torsion_Jx = 0.33*b*h^3 bending_Iy = b*h^3/12 bending_Iz = h*b^3/12
mass_density = rho xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m) joint_2_offset = (0*m, 0*m, 0*m);
add element # 2 type beam_elastic with nodes (2,3)
```

Figure 707.33: Elastic frame with beam_elastic elements.
cross_section = b*h  elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2
torsion_Jx = 0.33*b*h^3  bending_Iy = b*h^3/12  bending_Iz = h*b^3/12
mass_density = rho  xz_plane_vector = (1, 0, 1 )
  joint_1_offset = (0*m, 0*m, 0*m )  joint_2_offset = (0*m, 0*m, 0*m );
add element # 3 type beam_elastic with nodes (3,4)
cross_section = b*h  elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2
torsion_Jx = 0.33*b*h^3  bending_Iy = b*h^3/12  bending_Iz = h*b^3/12
mass_density = rho  xz_plane_vector = (1, 0, 1 )
  joint_1_offset = (0*m, 0*m, 0*m )  joint_2_offset = (0*m, 0*m, 0*m );
fix node #1 dofs all;
fix node #4 dofs all;
new loading stage "Fz";
add load # 1 to node # 2 type linear Fz=50*N;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
define load factor increment 1;
simulate 1 steps using static algorithm;
bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.10 27NodeBrick Cantilever Beam, Static Load

Problem description:

Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \). The force direction is shown in Figure (707.34).

![Figure 707.34: Problem description for cantilever beam.](image)

Numerical model:

The 27NodeBrick elements for cantilever beams is shown in Figure (707.35):

![Figure 707.35: 27NodeBrick elements for cantilever beams made of solid elements.](image)

ESSI model fei/DSL file:

```fei
model name "6meter_cantilever_27brick" ;
add material # 1 type linear_elastic_isotropic_3d
  mass_density = 0*kg/m^3
  elastic_modulus = 1e8*N/m^2
  poisson_ratio = 0.0;
add node # 1 at ( 0.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
add node # 2 at ( 0.00 *m, 0.00 *m, 0.00 *m) with 3 dofs;
```

10 add node # 3 at ( 6.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
11 add node # 4 at ( 5.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
12 add node # 5 at ( 4.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
13 add node # 6 at ( 3.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
... 
16 add node #117 at ( 5.50 *m, 0.50 *m, 1.00 *m) with 3 dofs;
17 
18 add element # 1 type 27NodeBrickLT with nodes( 2, 10, 8, 1, 15, 17, 28, 23, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47) use material # 1;
19 add element # 2 type 27NodeBrickLT with nodes( 10, 11, 7, 8, 17, 18, 27, 28, 48, 49, 50, 51, 52, 53, 34, 38, 54, 55, 56, 57, 58, 59, 43, 60, 61) use material # 1;
20 add element # 3 type 27NodeBrickLT with nodes( 11, 12, 6, 7, 18, 19, 26, 27, 62, 63, 64, 49, 65, 66, 67, 52, 54, 68, 69, 55, 70, 71, 72, 73, 58, 74, 75) use material # 1;
21 add element # 4 type 27NodeBrickLT with nodes( 12, 13, 5, 6, 19, 20, 25, 26, 76, 77, 78, 63, 79, 80, 81, 66, 68, 82, 83, 69, 84, 85, 86, 87, 72, 88, 89) use material # 1;
22 add element # 5 type 27NodeBrickLT with nodes( 13, 14, 4, 5, 20, 21, 24, 25, 90, 91, 92, 77, 93, 94, 95, 80, 82, 96, 97, 83, 98, 99, 100, 101, 86, 102, 103) use material # 1;
23 add element # 6 type 27NodeBrickLT with nodes( 14, 9, 3, 4, 21, 16, 22, 24, 104, 105, 106, 91, 107, 108, 109, 94, 96, 110, 111, 97, 112, 113, 114, 115, 100, 116, 117) use material # 1;
25 fix node # 1 dofs all;
26 fix node # 2 dofs all;
27 fix node # 15 dofs all;
28 fix node # 23 dofs all;
29 fix node # 32 dofs all;
30 fix node # 36 dofs all;
31 fix node # 37 dofs all;
32 fix node # 40 dofs all;
33 fix node # 45 dofs all;
35 new loading stage "Fz";
36 add load # 1 to node # 13 type linear Fz=2.777778*N;
37 add load # 2 to node # 24 type linear Fz=2.777778*N;
38 add load # 3 to node # 3 type linear Fz=2.777778*N;
39 add load # 4 to node # 34 type linear Fz=2.777778*N;
40 add load # 5 to node # 182 type linear Fz=11.111111*N;
41 add load # 6 to node # 177 type linear Fz=11.111111*N;
42 add load # 7 to node # 180 type linear Fz=11.111111*N;
43 add load # 8 to node # 183 type linear Fz=11.111111*N;
44 add load # 9 to node # 186 type linear Fz=44.444444*N;
46 define algorithm With_no_convergence_check;
47 define solver UMFPack;
48 define load factor increment 1;
simulate 1 steps using static algorithm;

bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.11 4NodeANDES Cantilever Beam, Force Perpendicular to Plane

Problem description:
Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, ν = 0.0.

![Cantilever beam diagram](image)

Figure 707.36: Cantilever beams

Numerical model:
For a force direction perpendicular to the plane, only the bending deformation is present.
The model is shown in Figure (707.37).

![Model diagram](image)

Figure 707.37: 4NodeANDES elements for cantilever beams under force perpendicular to plane.

**ESSI model fei/DSL file:**

```plaintext
model name "6meter_cantilever_4NodeANDES" ;
add material # 1 type linear_elastic_isotropic_3d
mass_density = 0*kg/m^3
elastic_modulus = 1e8*N/m^2
poisson_ratio = 0.0;
add node # 1 at ( 0.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at ( 6.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 3 at ( 1.0*m, 0.0*m, 0.0*m) with 6 dofs;
```

---

add node # 4 at (2.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 5 at (3.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 6 at (4.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 7 at (5.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 8 at (6.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 9 at (0.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 10 at (5.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 11 at (4.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 12 at (3.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 13 at (2.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 14 at (1.0*m, 1.0*m, 0.0*m) with 6 dofs;

h = 1*m;
add element # 1 type 4NodeShell_ANDES with nodes (1,3,14,9) use material # 1 ←
thickness = h;
add element # 2 type 4NodeShell_ANDES with nodes (3,4,13,14) use material # 1 ←
thickness = h;
add element # 3 type 4NodeShell_ANDES with nodes (4,5,12,13) use material # 1 ←
thickness = h;
add element # 4 type 4NodeShell_ANDES with nodes (5,6,11,12) use material # 1 ←
thickness = h;
add element # 5 type 4NodeShell_ANDES with nodes (6,7,10,11) use material # 1 ←
thickness = h;
add element # 6 type 4NodeShell_ANDES with nodes (7,2,8,10) use material # 1 ←
thickness = h;

fix node # 1 dofs all;
fix node # 9 dofs all;

new loading stage "Fz";
add load # 1 to node # 8 type linear Fz=50*N;
add load # 2 to node # 2 type linear Fz=50*N;

define algorithm With_no_convergence_check;
define solver ProfileSPD;
define load factor increment 1;
simulate 1 steps using static algorithm;
bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.12 4NodeANDES Cantilever Beams, In-Plane Force

Problem description:
Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.0$.

![Problem description for cantilever beams with in plane force](image)

**Numerical model:**

The 4NodeANDES elements under in-plane force is shown in Figure (707.39).

![4NodeANDES elements for cantilever beams under in-plane force](image)

**ESSI model fei/DSL file:**

```plaintext
1 model name "6meter_cantilever_4NodeANDES" ;
2 add material # 1 type linear_elastic_isotropic_3d
3   mass_density = 0*kg/m^3
4   elastic_modulus = 1e8*N/m^2
5   poisson_ratio = 0.0;
6
7 add node # 1 at ( 0.00*m, 0.00*m, 0.00*m) with 6 dofs;
8 add node # 2 at ( 6.00*m, 0.00*m, 0.00*m) with 6 dofs;
9 add node # 3 at ( 1.00*m, 0.00*m, 0.00*m) with 6 dofs;
10 add node # 4 at ( 2.00*m, 0.00*m, 0.00*m) with 6 dofs;
```

Figure 707.38: Problem description for cantilever beams with in plane force

Figure 707.39: 4NodeANDES elements for cantilever beams under in-plane force
add node # 5 at ( 3.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 6 at ( 4.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 7 at ( 5.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 8 at ( 6.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 9 at ( 0.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 10 at ( 5.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 11 at ( 4.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 12 at ( 3.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 13 at ( 2.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 14 at ( 1.00*m, 1.00*m, 0.00*m) with 6 dofs;
h = 1*m;
add element # 1 type 4NodeShell_ANDES with nodes (1,3,14,9) use material # 1
thickness = h
add element # 2 type 4NodeShell_ANDES with nodes (3,4,13,14) use material # 1
thickness = h
add element # 3 type 4NodeShell_ANDES with nodes (4,5,12,13) use material # 1
thickness = h
add element # 4 type 4NodeShell_ANDES with nodes (5,6,11,12) use material # 1
thickness = h
add element # 5 type 4NodeShell_ANDES with nodes (6,7,10,11) use material # 1
thickness = h
add element # 6 type 4NodeShell_ANDES with nodes (7,2,8,10) use material # 1
thickness = h
fix node # 1 dofs all;
fix node # 9 dofs all;
new loading stage "Fy";
add load # 1 to node # 8 type linear Fy=50*N;
add load # 2 to node # 2 type linear Fy=50*N;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
define load factor increment 1;
simulate 1 steps using static algorithm;
bye;

The ESSI model fei/DSL files for this example can be downloaded here.
## 707.13 27NodeBrick Cantilever Beams, Dynamic Input

**Problem description:**

Length=20m, Width=1m, Height=1m, E=504MPa, \( \nu = 0.4 \).

All degree of freedoms at the bottom nodes are fixed.

The load is a self weight with a dynamic displacement of supports.

![Problem description for one simple dynamic example](image)

**Numerical model:**

The numerical model applied 27NodeBrick to simulate the 1C (1 component) motion.

![Numerical model for one simple dynamic example](image)

**ESSI model fei/DSL file:**

```plaintext
1 model name "dynamic_example";
2 add material # 1 type linear_elastic_isotropic_3d_LT
3 mass_density = 2000*kg/m^3
```

elastic_modulus = 504000000.00*Pa
poisson_ratio = 0.4;

add node No 1 at (0*m, 0*m, 0*m) with 3 dofs;
add node No 2 at (0*m, 0.5*m, 0*m) with 3 dofs;
add node No 3 at (0*m, 1*m, 0*m) with 3 dofs;
add node No 4 at (0.5*m, 0*m, 0*m) with 3 dofs;
add node No 5 at (0.5*m, 0.5*m, 0*m) with 3 dofs;
add node No 6 at (0.5*m, 1*m, 0*m) with 3 dofs;

...  
...  
add node No 369 at (1*m, 1*m, 20*m) with 3 dofs;

add element # 1 type 27NodeBrickLT with nodes
(27,21,19,25,9,3,1,7,24,20,22,6,2,4,8,18,12,10,16,14,15,11,13,17,23,5) use material # 1;
add element # 2 type 27NodeBrickLT with nodes
(45,39,37,43,27,21,19,25,42,38,40,44,24,20,22,26,36,30,28,34,32,33,29,31,35,41,23) use material # 1;
add element # 3 type 27NodeBrickLT with nodes
(63,57,55,61,45,39,37,43,60,56,58,62,42,38,40,44,54,48,46,52,50,51,47,49,53,59,41) use material # 1;
add element # 4 type 27NodeBrickLT with nodes
(81,75,73,79,63,57,55,61,78,74,76,80,60,56,58,62,72,66,64,70,68,69,65,67,71,77,69) use material # 1;
add element # 5 type 27NodeBrickLT with nodes
(99,93,91,97,81,75,73,79,96,92,94,98,78,74,76,80,90,84,82,88,86,87,83,85,89,95,77) use material # 1;

...  
... 
add element # 20 type 27NodeBrickLT with nodes

add acceleration field # 1 ax = 0*g ay = 0*g az = -1*g;
add load # 1 to element # 1 type self_weight use acceleration field # 1;
add load # 2 to element # 2 type self_weight use acceleration field # 1;
add load # 3 to element # 3 type self_weight use acceleration field # 1;
add load # 4 to element # 4 type self_weight use acceleration field # 1;
add load # 5 to element # 5 type self_weight use acceleration field # 1;
add load # 6 to element # 6 type self_weight use acceleration field # 1;

...  
... 
add load # 20 to element # 20 type self_weight use acceleration field # 1;

fix node No 1 dofs uy uz;
fix node No 2 dofs uy uz;
fix node No 3 dofs uy uz;
fix node No 4 dofs uy uz;
fix node No 5 dofs uy uz;
fix node No 6 dofs uy uz;
fix node No 369 dofs uy uz;

zeta = 0.0166667;
fq1 = 3.75;
fq2 = 11.25;
omega1 = 2*pi*fq1;
omega2 = 2*pi*fq2;
zeta1 = zeta;
zeta2 = zeta;
alpha1 = \frac{2*\omega_1*\omega_2*(zeta_1*\omega_2-zeta_2*\omega_1)}{(\omega_2*\omega_2-\omega_1*\omega_1)};
beta1 = \frac{2*(zeta_2*\omega_2-zeta_1*\omega_1)}{(\omega_2*\omega_2-\omega_1*\omega_1)};
add damping # 1
  type Rayleigh
  with
  a0 = alpha1/s
  a1 = beta1*s
  stiffness_to_use = Initial_Stiffness;

add damping # 1 to element # 1;
add damping # 1 to element # 2;
add damping # 1 to element # 3;
add damping # 1 to element # 4;
add damping # 1 to element # 5;
add damping # 1 to element # 6;
...
add damping # 1 to element # 20;

new loading stage "impose_motion";

add imposed motion # 1001 to node # 1 dof ux
  displacement_scale_unit = 1*m displacement_file = "dis.txt"
  velocity_scale_unit = 1*m/s velocity_file = "vel.txt"
  acceleration_scale_unit = 1*m/s^2 acceleration_file = "acc.txt";

add imposed motion # 1002 to node # 2 dof ux
  displacement_scale_unit = 1*m displacement_file = "dis.txt"
  velocity_scale_unit = 1*m/s velocity_file = "vel.txt"
  acceleration_scale_unit = 1*m/s^2 acceleration_file = "acc.txt";

add imposed motion # 1003 to node # 3 dof ux
  displacement_scale_unit = 1*m displacement_file = "dis.txt"
  velocity_scale_unit = 1*m/s velocity_file = "vel.txt"
  acceleration_scale_unit = 1*m/s^2 acceleration_file = "acc.txt";
...
add imposed motion # 1009 to node # 9 dof ux
  displacement_scale_unit = 1*m displacement_file = "dis.txt"
velocity_scale_unit = 1*m/s velocity_file = "vel.txt"
acceleration_scale_unit = 1*m/s^2 acceleration_file = "acc.txt";

define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 50 steps using transient algorithm time_step = 0.005*s;

bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.14 4NodeANDES Square Plate, Four Edges Clamped

Problem description:

Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.3 \).

The four edges are clamped.

The load is a self weight.

Numerical model:

The element side length is 1 meter.

ESSI model fei/DSL file:

```plaintext
model name "square_plate";

add material # 1 type linear_elastic_isotropic_3d
mass_density = 1e2*kg/m^3 elastic_modulus = 1e8*N/m^2 poisson_ratio = 0.3;

add node # 1 at ( 0.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 2 at ( 20.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 3 at ( 1.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 4 at ( 2.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 5 at ( 3.00*m, 0.00*m, 0.00*m) with 6 dofs;
```

Figure 707.42: Square plate with four edges clamped
add node # 6 at (4.00*m, 0.00*m, 0.00*m) with 6 dofs;

...  

...  

add node # 441 at (19.00*m, 19.00*m, 0.00*m) with 6 dofs;

h = 1*m;

add element # 1 type 4NodeShell_ANDES with nodes(1, 3, 81, 80) use material # 1 thickness=h;
add element # 2 type 4NodeShell_ANDES with nodes(3, 4, 100, 81) use material # 1 thickness=h;
add element # 3 type 4NodeShell_ANDES with nodes(4, 5, 119, 100) use material # 1 thickness=h;
add element # 4 type 4NodeShell_ANDES with nodes(5, 6, 138, 119) use material # 1 thickness=h;
add element # 5 type 4NodeShell_ANDES with nodes(6, 7, 157, 138) use material # 1 thickness=h;
add element # 6 type 4NodeShell_ANDES with nodes(7, 8, 176, 157) use material # 1 thickness=h;

...  

...  

add element # 400 type 4NodeShell_ANDES with nodes(441, 41, 22, 43) use material # 1 thickness=h;

fix node # 1 dofs all;

fix node # 2 dofs all;
The ESSI model fei/DSL files for this example can be downloaded here.
707.15 One Dimensional DRM Model

Problem description:

A simple 1D DRM model is shown in Fig. (707.44). The "DRM element", "Exterior node" and "Boundary node" are required to be designated in the DRM HDF5 input. The format and script for the HDF5 input is available in DSL/input manual.

![Diagram of 1D DRM model](image)

Figure 707.44: 1D DRM model.

Numerical model:

**ESI model fei/DSL file:**

```plaintext
model name "DRM";

//Material for soil
add material # 1 type linear_elastic_isotropic_3d_LT
mass_density = 2000*kg/m^3
elastic_modulus = 1300*MPa
poisson_ratio = 0.3;

//Material for DRM layer
add material # 2 type linear_elastic_isotropic_3d_LT
mass_density = 2000*kg/m^3
elastic_modulus = 1300*MPa
poisson_ratio = 0.3;

//Material for exterior layer
add material # 3 type linear_elastic_isotropic_3d_LT
mass_density = 2000*kg/m^3
elastic_modulus = 1300*MPa
poisson_ratio = 0.3;
```

add node # 1 at (0.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 2 at (5.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 3 at (5.00*m, 5.00*m, 0.00*m) with 3 dofs;
add node # 4 at (0.00*m, 5.00*m, 0.00*m) with 3 dofs;
add node # 5 at (5.00*m, 0.00*m, 50.00*m) with 3 dofs;
add node # 6 at (5.00*m, 0.00*m, 5.00*m) with 3 dofs;
... 
add node # 52 at (0.00*m, 5.00*m, -5.00*m) with 3 dofs;

//
// add element # 1 type 8NodeBrickLT with nodes(1, 4, 3, 2, 24, 44, 34, 6) use ← material # 1;
add element # 2 type 8NodeBrickLT with nodes(24, 44, 34, 6, 23, 43, 33, 7) use ← material # 1;
... 
add element # 12 type 8NodeBrickLT with nodes(48, 47, 45, 46, 52, 51, 49, 50) ←
use material # 3;

//
fix node # 1 dofs uy;
fix node # 1 dofs uz;

Figure 707.45: 1D DRM model.
The ESSI model fei/DSL files for this example can be downloaded here.

The same model for this example with 27NodeBrickLT can be downloaded here.

**Long 1D DRM model 1000:1**

To show the wave propagation explicitly, a long 1D model (1000:1) similar to the 1D DRM model above was made in this section.

The model description is same to Fig.(707.44) except this model use far more soil elements.

The general view is shown in Fig.(707.46) below.

There is still now outgoing waves at the exterior layers, which is shown in Fig(707.47).

The ESSI model fei/DSL files for this example can be downloaded here.

The results can also be seen in this animation.
Figure 707.46: Long 1D DRM model
Figure 707.47: Long 1D DRM model: exterior layer
707.16 Three Dimensional DRM Model

Problem description:

As shown in Fig.(707.48), the DRM layer is used to add the earthquake motion.

![Diagram of 3D Domain Reduction Method example](image)

Figure 707.48: The diagram for 3D Domain Reduction Method example.

Numerical result:

ESSI model fei/DSL file:

```plaintext
model name "DRM" ;

//Material for soil
add material # 1 type linear_elastic_isotropic_3d_LT
  mass_density = 2000*kg/m^3
  elastic_modulus = 1300*MPa
  poisson_ratio = 0.3;

//Material for DRM layer
add material # 2 type linear_elastic_isotropic_3d_LT
  mass_density = 2000*kg/m^3
  elastic_modulus = 1300*MPa
  poisson_ratio = 0.3;
```

Material for exterior layer

add material # 3 type linear_elastic_isotropic_3d_LT
mass_density = 2000*kg/m^3
elastic_modulus = 1300*MPa
poisson_ratio = 0.3;

add node # 1 at (0.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 2 at (50.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 3 at (5.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 4 at (10.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 5 at (15.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 6 at (20.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 7 at (25.00*m, 0.00*m, 0.00*m) with 3 dofs;
add node # 2925 at (55.00*m, 55.00*m, -5.00*m) with 3 dofs;

add element # 1 type 8NodeBrickLT with nodes(1, 40, 41, 3, 150, 441, 603, 151) ←
The ESSI model fei/DSL files for this example can be downloaded [here](#).

The same model for this example with 27NodeBrickLT can be downloaded [here](#).
707.17 ShearBeam Element, Pisano Material

Problem description:

In the element type "ShearBeamLT", only one Gauss point exists. ShearBeamLT element was used here to test the Pisanó material model.

Vertical force $F_z$ was used to apply confinement to the element. Then, cyclic force $F_x$ is used to load point.

![Figure 707.50: ShearBeam element.](image)

Resulting stress-strain relationship is shown in Fig.(707.51).

ESSI model fei/DSL file:

```plaintext
def model name "pisanoLT";
add node # 1 at (0*m,0*m,0*m) with 3 dofs;
add node # 2 at (0*m,0*m,1*m) with 3 dofs;
fix node # 1 dofs all;
```

version: 28 May, 2021, 17:09
Figure 707.51: Shear stress-strain response.

```plaintext
7     fix node # 2 dofs uy;
8
9     add material # 1 type New_PisanoLT
10    mass_density = 2000*kg/m\(^3\)
11    elastic_modulus_1atm = 325*MPa poisson_ratio = 0.3
12    M_in = 1.4 kd_in = 0.0 xi_in = 0.0 h_in = 700 m_in = 0.7
13    initial_confining_stress = 0*kPa n_in = 0 a_in = 0.0 eplcum_cr_in = 1e-6;
14
15     add element # 1 type ShearBeamLT with nodes (1, 2) \ 
16        cross_section = 1*m\(^2\) use material # 1;
17
18     new loading stage "confinement";
19
20     add load # 1 to node # 2 type linear Fz = -200*kN;
21     define load factor increment 0.01;
22     define algorithm With_no_convergence_check ;
23     define solver UMFPack;
24     simulate 100 steps using static algorithm;
25
26     new loading stage "test01";
27     gamma_max = 3e-3;
28     add imposed motion # 2 to node # 2 dof ux
29     displacement_scale_unit = gamma_max*m displacement_file = "input_sine.txt"
30     velocity_scale_unit = gamma_max*m/s velocity_file = "input_sine.txt"
31     acceleration_scale_unit = gamma_max*m/s\(^2\) acceleration_file = "input_sine.txt";
32     define load factor increment 0.0005;
```
define algorithm With_no_convergence_check;
define solver UMFPack;
simulate 2000 steps using static algorithm;
bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.18 8NodeBrickLT Element, Drucker-Prager Material, Armstrong-Frederick Rotational Kinematic Hardening

Problem description:

This example is used to test the materials properties, such as $G/G_{\text{max}}$ against strains. The element type is 8NodeBrickLT. And there are two stages of loading. The first loading stage is confinement and the second loading stage is shearing.

The boundary condition is specially designed such that each Gauss point has the same stress state.

Results

Resulting stress-strain relationship is shown in Fig.(707.52).

![Shear stress-strain response](image_url)

Figure 707.52: Shear stress-strain response.

ESSI model fei/DSL file:

```plaintext
// Drucker Prager Armstrong Frederick
// This model is created by Jose.
model name "druckeraf";

// Parameters:
phi = 5;
ha = 1000;
cr = 973;
```
gam = 0.01;
Ncyc = 5;
Nsteps = 1000;
H = 1;
vp = 1000 * m/s;
vs = 500 * m/s;
rho = 2000 * kg/m^3;
p0 = 250 * kPa;
G = rho * vs^2;
M = rho * vp^2;
E = G * (3*M - 4*G) / (M - G);
nu = (M - 2*G) / (2*M - 2*G);
K0 = 1.0;
phirad = pi * phi / 180;
M = 6 * sin(phirad) / (3 - sin(phirad));

// Define the material:
add material # 1 type DruckerPragerArmstrongFrederickLT
  mass_density = 0 * kg/m^3
  elastic_modulus = E
  poisson_ratio = nu
  druckerprager_k = M
  armstrong_frederick_ha = ha * Pa
  armstrong_frederick_cr = cr * Pa
  isotropic_hardening_rate = 0 * E
  initial_confining_stress = 1 * Pa;

// define the node:
add node # 1 at (0*m, 0*m, 1*m) with 3 dofs;
add node # 2 at (1*m, 0*m, 1*m) with 3 dofs;
add node # 3 at (1*m, 1*m, 1*m) with 3 dofs;
add node # 4 at (0*m, 1*m, 1*m) with 3 dofs;
add node # 5 at (0*m, 0*m, 0*m) with 3 dofs;
add node # 6 at (1*m, 0*m, 0*m) with 3 dofs;
add node # 7 at (1*m, 1*m, 0*m) with 3 dofs;
add node # 8 at (0*m, 1*m, 0*m) with 3 dofs;

// add equal degree of freedom in three directions
add constraint equal dof with master node # 2 and slave node # 3 dof to constrain ux;
add constraint equal dof with master node # 2 and slave node # 6 dof to constrain ux;
add constraint equal dof with master node # 2 and slave node # 7 dof to constrain ux;
add constraint equal dof with master node # 3 and slave node # 4 dof to constrain uy;
add constraint equal dof with master node # 3 and slave node # 8 dof to restrain uy;
constrain uy;
add constraint equal dof with master node # 3 and slave node # 7 dof to constrain uy;

add constraint equal dof with master node # 1 and slave node # 2 dof to constrain uz;
add constraint equal dof with master node # 1 and slave node # 3 dof to constrain uz;
add constraint equal dof with master node # 1 and slave node # 4 dof to constrain uz;

// Define the element.
add element # 1 type 8NodeBrickLT with nodes (1, 2,3 , 4, 5, 6,7, 8) use material # 1;

new loading stage "confinement";
fix node # 1 dofs ux uy;
fix node # 2 dofs uy;
fix node # 4 dofs ux;

fix node # 5 dofs ux uy uz;
fix node # 6 dofs uy uz;
fix node # 7 dofs uz;
fix node # 8 dofs ux uz;

sigma_z = -3*p0/(1+2*K0);
sigma_x = K0*sigma_z;
sigma_y = K0*sigma_z;

//Z-face
add load # 1 to node # 1 type linear Fz = sigma_z*m^2/4;
add load # 2 to node # 2 type linear Fz = sigma_z*m^2/4;
add load # 3 to node # 3 type linear Fz = sigma_z*m^2/4;
add load # 4 to node # 4 type linear Fz = sigma_z*m^2/4;

//X-face
add load # 5 to node # 2 type linear Fx = sigma_x*m^2/4;
add load # 6 to node # 6 type linear Fx = sigma_x*m^2/4;
add load # 7 to node # 7 type linear Fx = sigma_x*m^2/4;
add load # 8 to node # 3 type linear Fx = sigma_x*m^2/4;
add load # 9 to node # 3 type linear Fy = sigma_y*m^2/4;
add load # 10 to node # 7 type linear Fy = sigma_y*m^2/4;
add load # 11 to node # 8 type linear Fy = sigma_y*m^2/4;
add load # 12 to node # 4 type linear Fy = sigma_y*m^2/4;

Nsteps_static=100;
define load factor increment 1/Nsteps_static;
define solver UMFPack;
define convergence test Norm_Displacement_Increment
tolerance = 1e-6
maximum_iterations = 100
verbose_level = 4;
define algorithm Newton;

define NDMaterialLT constitutive integration algorithm Euler_One_Step
  yield_function_relative_tolerance = 0.002
  stress_relative_tolerance = 0.002
  maximum_iterations = 1000;
simulate Nsteps_static steps using static algorithm;

new loading stage "shearing";
compute reaction forces;
add load # 13 to node # 1 type from_reactions;
add load # 14 to node # 4 type from_reactions;
free node # 1 dofs ux;
free node # 4 dofs ux;
fix node # 3 dofs uy;
fix node # 6 dofs ux;
fix node # 7 dofs ux uy;
fix node # 8 dofs uy;
add constraint equal dof with master node # 1 and slave node # 3 dof to constrain ux;
add constraint equal dof with master node # 1 and slave node # 4 dof to constrain ux;
add constraint equal dof with master node # 1 and slave node # 2 dof to constrain ux;
remove constraint equaldof node # 6;
remove constraint equaldof node # 7;
remove constraint equaldof node # 8;
n = 1;
while(n<=1)
{
  add load # 14+n to node # n type path_time_series
    Fx = 170.*kN
    series_file = "path.txt";
    n+=1;
}
define load factor increment 1/Nsteps;
define solver UMFPack;
define convergence test Norm_Displacement_Increment
tolerance = 1e-5
maximum_iterations = 100
verbose_level = 4;
define algorithm Newton;

define NDMaterialLT constitutive integration algorithm Euler_One_Step
    yield_function_relative_tolerance = 0.0002
    stress_relative_tolerance = 0.002
    maximum_iterations = 1000;

simulate Ncyc*Nsteps steps using static algorithm;

bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.19 Contact Element Under Static Loading

Two Bar Normal Contact Problem Under Monotonic Loading.

This is an example of normal monotonic loading on a 1-D contact/interface between two bars separated by an initial gap of 0.1 unit. An illustrative diagram of the problem statement is shown below.

Figure 707.53: Illustration of Two Bar Normal Contact Problem under monotonic loading with initial gap

ESSI model fei/DSL file:

```plaintext
model name "Two_Bar_Contact_Under_Normal_Monotonic>Loading" ;

// Adding material
add material #1 type uniaxial_elastic elastic_modulus = 1*Pa ➔
    viscoelastic_modulus = 0*Pa*s;

// Adding Nodes
add node #1 at (0*m,0*m,0*m) with 3 dofs;
add node #2 at (1*m,0*m,0*m) with 3 dofs;
add node #3 at (1.1*m,0*m,0*m) with 3 dofs;
add node #4 at (2.1*m,0*m,0*m) with 3 dofs;

// Adding Fixities
fix node #1 dofs ux uy uz;
fix node #4 dofs ux uy uz;
fix node #2 dofs uy uz ;
fix node #3 dofs uy uz ;

// Adding Truss Elements
add element #1 type truss with nodes (1,2) use material # 1 cross_section = ➔
    1*m^2 mass_density = 1*kg/m^3;
add element #2 type truss with nodes (3,4) use material # 1 cross_section = ➔
    1*m^2 mass_density = 1*kg/m^3;

// Adding Contact Element
add element #3 type FrictionalPenaltyContact with nodes (2,3)
    normal_stiffness = 1e10*N/m
    tangential_stiffness = 1e10*Pa*m
    normal_damping = 0*kN/m*s
    tangential_damping = 0*kN/m*s
    friction_ratio = 0.3
```

Jeremić et al., University of California, Davis version: 28. May, 2021, 17:09
contact_plane_vector = (1,0,0);

new loading stage "Adding_Normal_Load";

add load #1 to node #2 type linear Fx = 0.3*N;

Nsteps = 10;

tol = 5e-12;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 10
verbose_level = 4;

define algorithm Newton;
define solver UMFPack;

define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

bye;

The displacement output of Node 2 and Node 3 are shown below.

![Displacement output of Node 2 and Node 3](image)

Figure 707.54: Displacement of Nodes 2 and 3

The ESSI model fei/DSL files for this example can be downloaded [here](link).
707.20 Four Bar Contact Problem With Normal and Shear Force Under Monotonic Loading

This is an example to show the normal and tangential behaviour (stick and slip case) of contacts/interfaces using four bars in 2-D plane. The bars in x-directions are in contact (initial gap=0).

Figure 707.55: Illustration of Four Bar Normal Contact Problem With Normal and Shear Force Under Monotonic Loading with no initial gap

ESSI model fei/DSL file:
model name "Four_Bar_Contact_Under_Monotonic_Normal_and_Shear_Loading";

// Adding material
add material #1 type uniaxial_elastic elastic_modulus = 1*Pa
   viscoelastic_modulus = 0*Pa*s;

// Adding Nodes
add node #1 at (0*m,0*m,0*m) with 3 dofs;
add node #2 at (1*m,0*m,0*m) with 3 dofs;
add node #3 at (1*m,0*m,0*m) with 3 dofs;
add node #4 at (2*m,0*m,0*m) with 3 dofs;
add node #5 at (1*m,-1*m,0*m) with 3 dofs;
add node #6 at (1*m,1*m,0*m) with 3 dofs;

// Adding Truss Elements
add element #1 type truss with nodes (1,2) use material # 1 cross_section =
   1*m^2 mass_density = 1*kg/m^3;
add element #2 type truss with nodes (3,4) use material # 1 cross_section =
   1*m^2 mass_density = 1*kg/m^3;
add element #3 type truss with nodes (3,5) use material # 1 cross_section =
   1*m^2 mass_density = 1*kg/m^3;
add element #4 type truss with nodes (2,6) use material # 1 cross_section =
   1*m^2 mass_density = 1*kg/m^3;

// Adding Contact Element
add element #5 type FrictionalPenaltyContact with nodes (2,3)
   normal_stiffness = 1e12*N/m
   tangential_stiffness = 1e12*N/m
   normal_damping = 0*N/m*s
   tangential_damping = 0*N/m*s
   friction_ratio = 0.4
   contact_plane_vector = (1,0,0);

// Adding Fixities
fix node #1 dofs ux uy uz ;
fix node #4 dofs ux uy uz ;
fix node #5 dofs ux uy uz ;
fix node #6 dofs ux uy uz ;
fix node #2 dofs uz ;
fix node #3 dofs uz ;

new loading stage "Normal_Loading";

add load #1 to node #2 type linear Fx = 0.1*N;
tol = 1e-10;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 10
verbose_level = 4;
define algorithm Newton;
Nsteps= 10;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

new loading stage "Shear_Loading";

add load #2 to node #2 type linear Fy = 0.2*N;
tol = 1e-10;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 10
verbose_level = 4;

define algorithm Newton;
Nsteps= 100;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

bye;

The displacement output of Node 2 and Node 3 are shown below.
The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.56: Displacement of Nodes 2 and 3 along y direction
707.21  3-D Truss example with normal confinement and Shear Loading

A simple 3-D truss example with Normal confinement in z-direction of $F_N = 0.5N$, friction coefficient $\mu = 0.2$ and shear loading of magnitude $F_s = 0.5N$. Figure 707.57 below, shows the description of the problem.

Figure 707.57: Illustration of 3-D Truss Problem with confinement loading in z-direction of 0.5N and then shear loading of 0.5N in x-y plane

**ESSI model fei/DSL file:**

```plaintext
model name "3-D_Contact_Under_Normal_And_Tangential_Loading" ;

// Adding material
add material #1 type uniaxial_elastic elastic_modulus = 1*Pa ←
  viscoelastic_modulus = 0*Pa*s;

// Adding Nodes
add node #1 at (0*m,0*m,0*m) with 3 dofs; 
add node #2 at (0*m,0*m,0*m) with 3 dofs; 
add node #3 at (-1*m,0*m,0*m) with 3 dofs; 
add node #4 at (0*m,1*m,0*m) with 3 dofs; 
add node #5 at (0*m,0*m,1*m) with 3 dofs; 

// Adding Fixities
fix node #1 dofs ux uy uz; 
fix node #3 dofs ux uy uz; 
fix node #4 dofs ux uy uz; 
fix node #5 dofs ux uy uz; 

// Adding Truss Elements
add element #1 type truss with nodes (2,3) use material # 1 cross_section = ←
  1*m^2 mass_density = 1*kg/m^3;
add element #2 type truss with nodes (2,4) use material # 1 cross_section = ←
```

1*m^2 mass_density = 1*kg/m^3;

add element #3 type truss with nodes (2,5) use material # 1 cross_section = ←
1*m^2 mass_density = 1*kg/m^3;

// Adding Contact Element
add element #4 type FrictionalPenaltyContact with nodes (1,2)
normal_stiffness =1e10*N/m
tangential_stiffness = 1e10*Pa*m
normal_damping = 0*kN/m*s
tangential_damping = 0*kN/m*s
friction_ratio = 0.2
contact_plane_vector = (0,0,1);

new loading stage "Adding_Normal_Load";

add load #1 to node #2 type linear Fz = -0.5*N;
Nsteps = 1;
tol = 1e-10;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 1
verbose_level = 4;

define algorithm Newton;
define solver UMFPack;

define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

new loading stage "Shear_Loading";

add load #2 to node #2 type linear Fx = 0.4;
add load #3 to node #2 type linear Fy = 0.3;
tol = 1e-12;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 10
verbose_level = 4;

define algorithm Newton;

Nsteps= 20;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

bye;
The generalized displacement response of the tangential loading stage is shown below.

![Graphs showing displacement and force responses](image)

**Figure 707.58:** Displacements of Node 2 with applied shear tangential load step.

![Graphs showing force responses](image)

**Figure 707.59:** Resisting force by the contact/interface element with applied shear tangential load step.

The ESSI model fei/DSL files for this example can be downloaded here.
707.22 Six Solid Blocks Example With Contact

This is a 3-D solid block example with initial normal and then tangential load on different surfaces as shown below.

Figure 707.60: Illustration of Six Solid Blocks Example with Contact having first normal and then tangential loading stages.

ESSI model fei/DSL file:

```plaintext
model name "Six_Solid_Blocks_Example_With_Contact";

// Adding material
add material #1 type linear_elastic_isotropic_3d_LT mass_density=2000*kg/m^3 ←
elastic_modulus=200*MPa poisson_ratio=0.3;

// Adding Nodes
add node # 1 at (-1.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 2 at (-1.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 3 at (1.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 4 at (1.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 5 at (-1.500000*m,-0.500000*m,-2.000000*m) with 3 dofs;
add node # 6 at (-1.500000*m,0.500000*m,-2.000000*m) with 3 dofs;
add node # 7 at (0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 8 at (0.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 9 at (-0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 10 at (0.500000*m,-0.500000*m,-2.000000*m) with 3 dofs;
add node # 11 at (-0.500000*m,0.500000*m,-2.000000*m) with 3 dofs;
add node # 12 at (0.500000*m,0.500000*m,-2.000000*m) with 3 dofs;
add node # 13 at (-0.500000*m,0.500000*m,-2.000000*m) with 3 dofs;
```

add node # 14 at (0.500000*m,0.500000*m,-2.000000*m) with 3 dofs;
add node # 15 at (0.500000*m,-0.500000*m,-2.000000*m) with 3 dofs;
add node # 16 at (-0.500000*m,-0.500000*m,-2.000000*m) with 3 dofs;
add node # 17 at (-1.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 18 at (-1.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 19 at (1.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 20 at (1.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 21 at (-0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 22 at (0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 23 at (-0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 24 at (0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 25 at (-0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 26 at (0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 27 at (-0.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 28 at (0.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 29 at (-0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 30 at (0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 31 at (-0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 32 at (0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;

// Adding Solid 8 Node Brick Elements
add element #1 type 8NodeBrickLT with nodes (21,23,17,18,11,9,1,2) use ← material #1;
add element #2 type 8NodeBrickLT with nodes (13,16,5,6,21,23,17,18) use ← material #1;
add element #3 type 8NodeBrickLT with nodes (30,32,31,29,28,26,25,27) use ← material #1;
add element #4 type 8NodeBrickLT with nodes (14,15,16,13,22,24,23,21) use ← material #1;
add element #5 type 8NodeBrickLT with nodes (19,20,24,22,4,3,10,12) use ← material #1;
add element #6 type 8NodeBrickLT with nodes (7,8,15,14,19,20,24,22) use ← material #1;

// Adding some variables
Kn = 1e12*N/m; // normal penalty stiffness
Kt = 1e12*N/m; // tangential penalty stiffness
Cn = 0*N/m*s; // normal penalty damping
Ct = 0*N/m*s; // tangential penalty damping
nu = 0.4; // friction ratio

// Adding Contact Element
add element #7 type FrictionalPenaltyContact with nodes (9,25)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (1,0,0);
add element #8 type FrictionalPenaltyContact with nodes (10,26)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (-1,0,0);

add element #9 type FrictionalPenaltyContact with nodes (11,27)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (1,0,0);

add element #10 type FrictionalPenaltyContact with nodes (12,28)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (-1,0,0);

add element #11 type FrictionalPenaltyContact with nodes (21,29)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (1,0,0);

add element #12 type FrictionalPenaltyContact with nodes (22,30)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (-1,0,0);

add element #13 type FrictionalPenaltyContact with nodes (23,31)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (1,0,0);

add element #14 type FrictionalPenaltyContact with nodes (24,32)
  normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (-1,0,0);

add element #15 type FrictionalPenaltyContact with nodes (21,29)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

add element #16 type FrictionalPenaltyContact with nodes (22,30)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

add element #17 type FrictionalPenaltyContact with nodes (23,31)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

add element #18 type FrictionalPenaltyContact with nodes (24,32)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

// Adding Fixities
fix node #5 dofs ux uy uz;
fix node #6 dofs ux uy uz;
fix node #13 dofs ux uy uz;
fix node #16 dofs ux uy uz;
fix node #15 dofs ux uy uz;
fix node #14 dofs ux uy uz;
fix node #7 dofs ux uy uz;
fix node #8 dofs ux uy uz;
fix node #17 dofs ux uy;
fix node #18 dofs ux uy;
fix node #1 dofs ux uy;
fix node #2 dofs ux uy;
fix node #20 dofs ux uy;
fix node #19 dofs ux uy;
fix node #3 dofs ux uy;
fix node #4 dofs ux uy;
fix node #9 dofs uy;
fix node #10 dofs uy;
fix node #23 dofs uy;
fix node #24 dofs uy;
fix node #11 dofs uy;
fix node #21 dofs uy;
fix node #12 dofs uy;
fix node #22 dofs uy;
fix node #25 dofs uy;
fix node #26 dofs uy;
fix node #27 dofs uy;
fix node #28 dofs uy;
fix node #29 dofs uy;
fix node #30 dofs uy;
fix node #31 dofs uy;
fix node #32 dofs uy;

new loading stage "Normal_Loading";

add load #1 to element #3 type surface at nodes (25,26,27,28) with magnitude \(-1\text{ Pa}\);
tol = 1e-12;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 100
verbose_level = 4;

define algorithm Newton;
Nsteps= 10;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

new loading stage "Shear_Loading";

add load #2 to element #3 type surface at nodes (26,28,30,32) with magnitude \(-1\text{ Pa}\);
tol = 1e-12;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 100
verbose_level = 4;

define algorithm Newton;
Nsteps = 10;
define solver UMFPack;
define load factor increment $1/Nsteps$;
simulate $Nsteps$ steps using static algorithm;
bye;

The generalized displacement field of the two loading stages normal loading and tangential loading is shown below.

![Figure 707.61: Generalized displacement magnitude visualization of normal loading](image1)

![Figure 707.62: Generalized displacement magnitude visualization of tangential loading](image2)
The ESSI model fei/DSL files for this example can be downloaded here.
707.23 Pure shear model for G/Gmax plot

Problem description:

The pure shear model for G/Gmax plot

![Diagram of pure shear model for confinement and shearing](image)

Figure 707.63: The pure shear model for (a) confinement and (b) shearing

ESSI model fei/DSL file:

```plaintext
model name "GGmax" ;
// Parameters:
phi = 0.0135713590083;
ha = 2.94767923453;
cr = 1854.31984573;
rho=1922.5 ;
deepth=0.1524/2;
confinstress=9.8*depth*rho;
G=12388.33;
p0 = confinstress*Pa;
phirad = pi*phi/180;
M = 6*sin(phirad)/(3-sin(phirad));
nu=0.3;
add material # 1 type DruckerPragerArmstrongFrederickLT
mass_density = rho*kg/m^3
elastic_modulus = 2*G*(1+nu)*Pa
poisson_ratio = nu
druckerprager_k = M
armstrong_frederick_ha = ha*Pa
armstrong_frederick_cr = cr*Pa
```

Jeremić et al., University of California, Davis
isotropic_hardening_rate = 0*Pa
initial_confining_stress = 10*Pa;
add node # 1 at ( 1.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 2 at ( 0.0000 *m, 1.0000 *m, 0.0000 *m) with 3 dofs;
add node # 3 at ( 1.0000 *m, 2.0000 *m, 0.0000 *m) with 3 dofs;
add node # 4 at ( 2.0000 *m, 1.0000 *m, 0.0000 *m) with 3 dofs;
add node # 5 at ( 1.0000 *m, 0.0000 *m, 1.0000 *m) with 3 dofs;
add node # 6 at ( 0.0000 *m, 1.0000 *m, 1.0000 *m) with 3 dofs;
add node # 7 at ( 1.0000 *m, 2.0000 *m, 1.0000 *m) with 3 dofs;
add node # 8 at ( 2.0000 *m, 1.0000 *m, 1.0000 *m) with 3 dofs;
add element # 1 type 8NodeBrickLT with nodes(1,2,3,4,5,6,7,8) use material # 1;
// fix the y direction for node 2,4,6,8
fix node # 2 dofs uy ;
fix node # 4 dofs uy ;
fix node # 6 dofs uy ;
fix node # 8 dofs uy ;
// fix the x direction for node 1,3,5,7
fix node # 1 dofs ux ;
fix node # 3 dofs ux ;
fix node # 5 dofs ux ;
fix node # 7 dofs ux ;
// Stage 1: confinement
new loading stage "confinement";
add load # 1 to node # 1 type linear Fy= p0*m^2;
add load # 2 to node # 3 type linear Fy= - p0*m^2;
add load # 3 to node # 5 type linear Fy= p0*m^2;
add load # 4 to node # 7 type linear Fy= - p0*m^2;
add load # 5 to node # 2 type linear Fx= p0*m^2;
add load # 6 to node # 4 type linear Fx= - p0*m^2;
add load # 7 to node # 6 type linear Fx= p0*m^2;
add load # 8 to node # 8 type linear Fx= - p0*m^2;
// confinement at z direction
add load # 101 to node # 1 type linear Fz= p0*m^2;
add load # 102 to node # 2 type linear Fz= p0*m^2;
add load # 103 to node # 3 type linear Fz= p0*m^2;
add load # 104 to node # 4 type linear Fz= p0*m^2;
add load # 105 to node # 5 type linear Fz= - p0*m^2;
add load # 106 to node # 6 type linear Fz= - p0*m^2;
add load # 107 to node # 7 type linear Fz= - p0*m^2;
add load # 108 to node # 8 type linear Fz= - p0*m^2;
// add algorithm and solver
Nsteps=100;
define load factor increment 1/Nsteps;
define solver ProfileSPD;
define convergence test Norm_Displacement_Increment
tolerance = 1e-5
maximum_iterations = 100
verbose_level = 4;
// define algorithm With_no_convergence_check;
define algorithm Newton;
define NDMaterialLT constitutive integration algorithm Euler_One_Step
yield_function_relative_tolerance = 0.00002
stress_relative_tolerance = 0.0002
maximum_iterations = 1000;
simulate Nsteps steps using static algorithm;
// ------------------------------------------------------------------------
// Stage 2: shear
new loading stage "shear";
// fix all the uz, since we want plane strain.
i=1;
while (i<9) {
    remove load # 100+i;
    fix node # i dofs uz;
i=i+1;
};
shearforce=1.6*kN;

add load # 9 to node # 1 type linear Fy= shearforce;// series_file = "path.txt";
add load # 10 to node # 3 type linear Fy=-shearforce;// series_file = "path.txt";
add load # 11 to node # 5 type linear Fy= shearforce;// series_file = "path.txt";
add load # 12 to node # 7 type linear Fy=-shearforce;// series_file = "path.txt";
add load # 13 to node # 2 type linear Fx=-shearforce;// series_file = "path.txt";
add load # 14 to node # 4 type linear Fx= shearforce;// series_file = "path.txt";
add load # 15 to node # 6 type linear Fx=-shearforce;// series_file = "path.txt";
add load # 16 to node # 8 type linear Fx= shearforce;// series_file = "path.txt";

// add algorithm and solver
Nsteps=1e4;
define static integrator displacement_control using node # 1 dof uy increment = 1e-2/Nsteps*m;
define convergence test Norm_Displacement_Increment tolerance = 0.000001 
maximum_iterations = 100 verbose_level = 0;
define solver ProfileSPD;
define algorithm Newton;
define NDMaterialLT constitutive integration algorithm Euler_One_Step
yield_function_relative_tolerance = 0.00002
stress_relative_tolerance = 0.0002
maximum_iterations = 1000;
simulate Nsteps steps using static algorithm; bye;

Figure 707.64: The G/Gmax results

The ESSI model fei/DSL files for this example can be downloaded here.
707.24 Multi-yield-surface von-Mises for G/Gmax plot

Problem description:

This model illustrates the G/Gmax input to multi-yield-surface von-Mises material. This example is based on one Gauss-point with multi-yield-surface von-Mises material. The G/Gmax is converted to material modeling parameters (yield-surface size and hardening parameter) inside the DSL.

ESSI model fei/DSL file:

```plaintext
model name "GGmax";
add material # 1 type vonMisesMultipleYieldSurfaceGoverGmax
  mass_density = 0.0*kg/m^3
  initial_shear_modulus = 3E8 * Pa
  poisson_ratio = 0.0
  total_number_of_shear_modulus = 9
  GoverGmax =
    "1,0.995,0.966,0.873,0.787,0.632,0.109,0.320,0.063"
  ShearStrainGamma =
    "0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01"
;
incr_size = 0.000001 ;
max_strain= 0.005 ;
num_of_increm = max_strain/incr_size -1 ;
simulate constitutive testing strain control pure shear use material # 1
  confinement_strain = 0.0
  strain_increment_size = incr_size
  maximum_strain = max_strain
  number_of_increment = num_of_increm;
bye;
```

Computed G/Gmax curve exactly matches the one used for input at control points.

The difference in G/Gmax between control points can be reduced by using more than just 9 control points as in this example.
Figure 707.65: Stress-Strain Relationship

Figure 707.66: The G/Gmax results.
Figure 707.67: Damping Ratio Plot
707.25 Multi-yield-surface Drucker-Prager for G/Gmax plot

Problem description:
This model illustrates the G/Gmax input to multi-yield-surface Drucker-Prager material. Purely
deviatoric plastic flow is used in this material, which means that the parameter dilation\_scale is set
to zero. If user wants to model change of volume (dilation or compression) for this material, then
G/Gmax curve need to be iterated upon manually by changing yield surface size directly, which is done
using different DruckerPrager\_MultipleYieldSurface command. This example is based on one Gauss-point
which use multi-yield-surface Drucker-Prager material. The G/Gmax is converted to the yield-surface
size and hardening parameter inside the DSL.

ESI model fei/DSL file:

```plaintext
model name "G/Gmax";
add material # 1 type DruckerPrager\_MultipleYieldSurfaceGoverGmax
  mass_density = 0.0*kg/m^3
  initial_shear_modulus = 3E8 * Pa
  poisson_ratio = 0.0
  initial_confining_stress = 1E5 * Pa
  reference_pressure = 1E5 * Pa
  pressure_exponential_n = 0.5
  cohesion = 0.0 * Pa
  dilation\_angle_\_eta = 1.0
  dilation\_scale = 0.0
  total_number_of_shear_modulus = 9
  GoverGmax =
    "1,0.995,0.966,0.873,0.787,0.620,0.109,0.063"
  ShearStrainGamma =
    "0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01"
;
incr\_size = 0.000001 ;
max\_strain= 0.005 ;
num\_of\_increment = max\_strain/incr\_size -1 ;
simulate constitutive testing strain control pure shear use material # 1
  confinement\_strain = 0.0
  strain\_increment\_size = incr\_size
  maximum\_strain = max\_strain
  number\_of\_increment = num\_of\_increment;
bye;
```

Inside the DSL, the yield surface radius is calculated as $\sqrt{3}\sigma_y$, where $\sigma_y$ is the yield stress of the
对应的yield surface. Then, the radius is divided by the confinement to obtain the slope (opening angle).
The hardening parameter is calculated as

\[ \frac{1}{H'_i} = \frac{1}{H_i} - \frac{1}{2G} \]  

(707.1)

where \( H'_i \) is the current hardening parameter corresponding to yield surface \( i \). \( H_i \) is the current tangent shear modulus to surface \( i \), namely, \( H_i = 2\left(\tau_{i+1} - \tau_i\right)\). And \( G \) is the initial shear modulus.
Figure 707.69: Nested-Yield-Surface Drucker-Prager $G/G_{\text{max}}$ results

Figure 707.70: Damping Ratio Plot
Appendix 708

Brief History of the Real-ESSI Simulator Development

(1986-)
This section briefly describes history of the development of the Finite Element Interpreted, FEI, that is currently represented by the Real-ESSI Simulator system. Developments are presented chronologically, with very brief description of capabilities, and with references to further reading and documents with more information.

1986-1988: Development of the FRAME_and_GRID program, using BASIC programming language, on SHARP 1500, CASIO 1000 (48KB RAM) and ZX Spectrum (128KB RAM), by Boris Jeremić, undergraduate student at the University of Belgrade.

1988-1989: Development of the Earthquake Soil Structure Interaction (ESSI) Program in time domain for axisymmetric solids with general 3D loads, using higher modes of response in circumferential direction, expanded in Fourier series, so that any general 3D loading and deformation can be modeled, using FORTRAN programming language, on PC-DOS, x286+287, 640KB+384KB RAM, by Boris Jeremić, undergraduate student at the University of Belgrade, as part of his Diploma Thesis (Jeremić, 1989).

1989-1992: Development of the Finite Element Interpreter (FEI), a general purpose static and dynamic, elastic and elastic-plastic finite element program for solids, rudimentary parser for a simple Domain Specific Language (DSL), using C Programming language, on PC-DOS, x286+287, 640KB+384KB RAM, by Boris Jeremić, a staff engineer at (a) Energoprojekt-Hidroinžinjering Company in Belgrade, Yugoslavia, at (b) Bekhme Dam Project site in Iraq, and at (c) Gasser&Scepan Design Bureau in Baar, Switzerland.

1992-1997: Development of the program FEM, featuring small and large deformation elasto-plasticity, solids (bricks with 8, 20 and 27 nodes), solution advancement control, using C++ Programming language, on Sun-SparcStation 5, Solaris, 256MB RAM, and on PC-DOS x386, x486 and on PC-Linux-TurboRedHat, by Boris Jeremić, a graduate student at the University of Colorado at Boulder, as part of his Master Thesis (Jeremić, 1994) and PhD Dissertation (Jeremić, 1997).
1997-2000: Continued development of the program FEM, addition of dynamics from ESSI, structural elements from FRAME_and_GRID, Parallel version, MPI based, linking with FEI, using C++ Programming language, on PC-Linux, and PC-Linux cluster: NorthCountry, 4 nodes + master, 100-based T network, by Professor Boris Jeremić, at Clarkson University and at the University of California at Davis.

2000-2006: Developments continued with introduction of all the previous and new developments from FEM into G3 Framework, later renamed OpenSEES, at PEER, using C++ Programming language, on PC-Linux, by Professor Boris Jeremić and co-workers at the University of California at Davis, CA, USA, see Final Report Presentation.

2006-Present: Development of the Real ESSI Simulator System (aka NRC-ESSI, MS-ESSI), using C++, FORTRAN, FEI-DSL, Python Programming languages, on PC-Linux, by Professor Boris Jeremić and co-workers at UCD. For details see main Real-ESSI Simulator web site or real-essi.us or real-essi.info (they all point to the same URL).
Appendix 709

Work Organization

(1989.)
This section describes in some detail work organization related to the development of FEI modeling and computational system.

709.1 Communication

Tablets, smart phones, laptops and computers, using https://zoom.us/ as it works on linux and all other OSs.

709.2 Writing (Notes, Code, &c.) Version Control

709.2.1 Source Code

Memory Leaks Memory leaks are best discovered by running Valgrind (http://valgrind.org/).

There are a number of tools that can be used with Valgrind. Mentioned are some of the most important ones, with example commands:

- (time valgrind --tool=cachegrind $argv[1] > $argv[1].cachegrind.$TIMESTAMP.out)>&! $argv[1].cachegrind.$TIMESTAMP.err
- valgrind -v --leak-check=yes --show-reachable=yes --num-callers=32 --trace-malloc=yes --error-limit=no --tool=massif $argv[1]

Examples use syntax from few years ago, so should be proper syntax should be verified using excellent Valgrind documentation.
709.2.2 Lecture Notes

Maintain lecture notes using git on https://github.com/.

Checking all http links in lecture notes using script ESSI_check_URLs_in_lecture_notes.sh in bin.

709.2.3 Bibliography

Bibliography List.

Papers of interest are organized in bibtex files (managed through git version control.

A list of those paper is compiled and available at:


Bibliography Repository.

Most listed papers are available at:
http://sokocalo.engr.ucdavis.edu/~jeremic/PAPERSlocalREPO/. This site is only accessible to members of the Computational Mechanics group at University of California at Davis, and few other collaborating entities.
709.3 Backup

709.4 Calendar

709.5 Useful Programs and Scripts

709.5.1 Backup Scripts

709.5.2 Domain Reduction Method Processing Programs and Scripts

DRM Node Extraction for fk.

fk Output Processing for DRM.

709.5.3 Pre Processing Programs and Scripts

709.5.4 Post Processing Programs and Scripts

709.5.5 Parallel Computer Architecture

http://www.open-mpi.org/projects/hwloc/
Appendix 710

Collected Bibliography

Compilation of all collected bibliography, over years, not necessarily cited in this book.
Bibliography

by:

Jeremić CompMech Group
Department of Civil and Environmental Engineering
University of California, Davis
Bibliography


[Ala ] Fadel Alameddine. Private communications. ..., 2003 --.


[A.TXX] L. A.Taber. Application of shell theory to cardiac mechanics. XX, XX.


[BA08] D. M. Boore and G. M. Atkinson. Ground-motion prediction equations for the average horizontal component of pga, pgv, and 5 %-damped psa at spectral periods between 0.01 s and 10.0 s. Earthquake Spectra, March 2008.


Ross Boulanger. Personal communications, 1999-present.


Jeremić et al., Real-ESSI


Richard Felder and Rebecca Brent. ExCEEd effective college teaching seminar: Course material. 1999 Civil Engineering Conference and Exposition, Charlotte, North Carolina, October 16-17 1999.


[FD05] Gregory Fenves and Mathew Dryden. Nees sfsi demonstration project. NEES project meeting, TX, Austin, August 2005.


engineering student performance and retention. I success and failure of introductory courses. Journal of

and retention. V. comparisons with traditionally-taught students. Journal of Engineering Education,


[FFM+95] R. M Felder, G. N. Felder, M. Mauney, C. E. Hamrin Jr., and E. J. Dietz. A longitudinal study of
engineering student performance and retention. III gender differences in student performance and attitudes.


[FG96] Peter Fajfar and Peter Gašperšič. The N2 method for the seismic damage analysis of rc buildings.


subjected to reverse faulting: Analysis and experiments. Soil Dynamics and Earthquake Engineering,

[FHM95] Carlos A. Felippa, Bjørn Haugen, and Carmelo Militello. From the individual element test to finite element


In Panos Dakoulas, Mishac Yegian, and Robert D. Holtz, editors, Proceedings of a Specialty Conference:
Geotechnical Earthquake Engineering and Soil Dynamics III, Geotechnical Special Publication No. 75,

[Fin00] W. D. Liam Finn. State-of-the-art of geotechnical earthquake engineering practice. Soil Dynamics and

[Fin18] W.D. Liam Finn. Performance based design in geotechnical earthquake engineering. Soil Dynamics and


Qing He, Houle Gan, and Dan Jiao. An explicit time-domain finite-element method that is unconditionally stable. Purdue e-Pubs; ECE Technical Reports 421, Purdue University, 2011.


E. Hinton, T. K. Hellen, and L. P. R. Lyons. On elasto-plastic benchmark philosophies. ... pages 389–407, ...


[Hin11] Klaus-G. Hinzen. Rotation of vertically oriented objects during earthquakes. published via email to rotation@lists.geophysik.uni-muenchen.de group, July 2011.


[Hua03] Bor-Shouh Huang. Ground rotational motions of the 1999 Chi-Chi, Taiwan earthquake as inferred from dense array observations. *Geophysical Research Letters*, 30(6), 2003.


Boris Jeremić. Dynamic analysis of axisymmetric solids subjected to non-symmetric loading by the finite element method. Diploma thesis., Faculty of Civil Engineering, Belgrade University, July 1989. in Serbian. (Борис Jeremiћ Динамична Анализа Ротационо Симетричних Тела Оптерећених Несиметричним Оптерећенима Методом Коначних Еlemenата, Дипломска Рад Грађевински Факултет Универзитета у Београду.)


Nicolas M. Kuehn and Norman A. Abrahamson. Spatial correlations of ground motion for non-ergodic seismic hazard analysis. Earthquake Engineering & Structural Dynamics, 0(0):1–19, 2019.


F. Ostadan, N. Deng, and R. Kennedy. Soil structure interaction analysis including ground motion incoherency effects. *18th International Conference on Structural Mechanics in Reactor Technology (SMiRT 18), Beijing, China, SMiRT18-K04-7*, August 2005.


This page seems to be a page from a document containing a list of references. The references are properly formatted in the style of bibliographic citations. Each entry includes the name of the author, year of publication, and specific details such as the title of the work, journal name, volume, issue, pages, and publisher. The references cover a range of topics including earthquake engineering, soil mechanics, and computational mechanics, among others. The page also contains a small logo at the bottom, which appears to be related to the University of California, Davis, as indicated by the text "Real-ESSI Lecture Notes page: 3036 of 3091" and "version: 28. May, 2021, 17:09."


[Ric11] Lewis Fry Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, 210:307–357, 1911.


Real-ESSI Lecture Notes  page: 3050 of 3091


Jeremić et al., Real-ESSI


Travis Swatson. 3D data visualization and modeling with a haptic interface. UCDavis seminar slides, May 1998.


Han Yang, Wen-Jie Xu, Qi-Cheng Sun, and Yuan Feng. Study on the meso-structure development in direct shear tests of a granular material. *Powder Technology*, 2016.


