## Contents

1 **Input, Domain Specific Language (DSL)** | 8
1.1 Chapter Summary and Highlights | 9
1.2 Introduction | 9
1.3 Domain Specific Language (DSL), English Language Binding | 10
   1.3.1 Running Real-ESSI | 10
   1.3.2 Finalizing Real-ESSI Program Run | 12
   1.3.3 Real-ESSI Variables, Basic Units and Flow Control | 13
   1.3.4 Modeling | 16
      Modeling, Material Model: Adding a Material Model to the Finite Element Model | 18
      Modeling, Material Model: Linear Elastic Isotropic Material Model | 19
      Modeling, Material Model: Cross Anisotropic Linear Elastic Material Model | 20
      Modeling, Material Model: von Mises Associated Material Model with Linear Isotropic and/or Kinematic Hardening | 21
      Modeling, Material Model: von Mises Associated Material Model with Isotropic Hardening and/or Armstrong-Frederic Nonlinear Kinematic Hardening | 22
      Modeling, Material Model: Drucker-Prager Associated Material Model with Linear Isotropic and/or Kinematic Hardening | 23
      Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederick Nonlinear Kinematic Hardening | 24
      Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederick Nonlinear Kinematic Hardening and Nonlinear Duncan-Chang Elasticity | 25
      Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Kinematic Hardening | 26
      Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Armstrong-Frederick Nonlinear Kinematic Hardening | 27
Modeling, Material Model: Rounded Mohr-Coulomb Associated Linear Isotropic Hardening Material Model

Modeling, Material Model: Cam Clay Material Model

Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model

Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model that Matches $G/G_{max}$ Curves

Modeling, Material Model: Drucker-Prager Nonassociated Multi-Yield Surface Material Model

Modeling, Material Model: Drucker-Prager Nonassociated Material Model that Matches $G/G_{max}$ Curves

Modeling, Material Model: Rounder Mohr-Coulomb Nonassociated Multi-Yield Surface Material Model

Modeling, Material Model: Tsinghua Liquefaction Material Model

Modeling, Material Model: SANISand Material Model, version 2004

Modeling, Material Model: SANISand Material Model, version 2008

Modeling, Material Model: Cosserat Linear Elastic Material Model

Modeling, Material Model: von Mises Cosserat Material Model

Modeling, Material Model: Uniaxial Linear Elastic, Fiber Material Model

Modeling, Material Model: Plane Stress Layered Material

Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 02

Modeling, Material Model: Faria-Oliver-Cervera Concrete Material

Modeling, Material Model: Plane Stress Plastic Damage Concrete Material

Modeling, Material Model: Plane Stress Rebar Material

Modeling, Nodes: Adding Nodes

Modeling, Nodes: Define Nodal Physical Group

Modeling, Nodes: Adding Nodes to Nodal Physical Group

Modeling, Nodes: Removing Nodal Physical Group

Modeling, Nodes: Print Nodal Physical Group

Modeling, Nodes: Removing Nodes

Modeling, Nodes: Adding Nodal Mass, for 3DOFs and/or 6DOFs

Modeling, Finite Element: Adding Finite Elements

Modeling, Finite Element: Define Finite Element Physical Group
Modeling, Finite Element: Adding Elements to Physical Element Group ............... 65
Modeling, Finite Element: Remove Physical Finite Element Group .................. 66
Modeling, Finite Element: Print Physical Finite Element Group ...................... 67
Modeling, Finite Element: Remove Finite Element ...................................... 68
Modeling, Finite Element: Truss Element ................................................. 69
Modeling, Finite Element: Shear Beam Element ........................................ 70
Modeling, Finite Element: Elastic Beam–Column Element ............................. 71
Modeling, Finite Element: Timoshenko Elastic Beam–Column Element .......... 73
Modeling, Finite Element: Adding 1D Fiber to a Beam Cross Section ............. 75
Modeling, Finite Element: Adding Fiber Section to the Finite Element Model ... 76
Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element . 77
Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element with Corotational Coordinate Transformation .............................................. 78
Modeling, Finite Element: 3DOF+6DOF=9DOF Beam-Column Element .......... 79
Modeling, Finite Element: 4 Node ANDES Shell with Drilling DOFs ............... 81
Modeling, Finite Element: 3 Node ANDES Shell with Drilling DOFs ............... 82
Modeling, Finite Element: 4 Node Shell NLDKGQ, or 4 Node Shell Xin-Zheng-Lu . 83
Modeling, Finite Element: Inelastic Layered Shell Section ............................ 84
Modeling, Finite Element: ElasticMembranePlaneStress Element (to be removed!) . 85
Modeling, Finite Element: InelasticMembranePlaneStress Element (to be removed!) . 86
Modeling, Finite Element: SuperElementLinearElasticImport ........................ 87
Modeling, Finite Element: 8 Node Brick Element ...................................... 89
Modeling, Finite Element: 20 Node Brick Element .................................. 91
Modeling, Finite Element: 27 Node Brick Element .................................. 93
Modeling, Finite Element: Variable 8-27 Node Brick Element ...................... 95
Modeling, Finite Element: 8 Node Brick u-p Element ................................ 97
Modeling, Finite Element: 20 Node Brick u-p Element ............................. 100
Modeling, Finite Element: 27 Node Brick u-p Element ............................. 103
Modeling, Finite Element: 8 Node Brick u-p-U Element ............................ 106
Modeling, Finite Element: 20 Node Brick u-p-U Element ............................ 109
Modeling, Finite Element: 27 Node Brick u-p-U Element ............................ 112
Modeling, Finite Element: 8 Node Cosserat Brick Element .......................... 115
Modeling, Finite Element: Bonded Contact/Interface/Joint Element ............... 116
Modeling, Finite Element: Force Based Dry Hard Contact/Interface/Joint Element ... 117
Modeling, Finite Element: Force Based Dry Soft Contact/Interface/Joint Element ... 118
Modeling, Finite Element: Force Based Coupled Hard Contact/Interface/Joint Element 120
Modeling, Finite Element: Force Based Coupled Soft Contact/Interface/Joint Element 121
Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior 123
Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior 125
Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior 127
Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior 130
Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior 132
Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior 134
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior 137
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior 139
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior 141
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior 144
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior 146
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior 148
Modeling, Finite Element: Neoprene Isolator Finite Element 151
Modeling, Finite Element: Lead Core Rubber Isolator/Dissipator Element 152
Modeling, Finite Element: Frictional Pendulum Isolator/Dissipator Finite Element version01 153
Modeling, Finite Element: Frictional Pendulum Isolator/Dissipator Finite Element version03 154
Modeling, Damping: Adding Rayleigh Damping 155
Modeling, Damping: Adding 3rd Order Caughey Damping 156
Modeling, Damping: Adding 4th Caughey Damping 157
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation: Dynamic Solution Advancement with the Constant Time Step</td>
<td>201</td>
</tr>
<tr>
<td>Simulation: Dynamic Solution Advancement with Variable Time Step</td>
<td>202</td>
</tr>
<tr>
<td>Simulation: Generalized Eigenvalue Analysis</td>
<td>203</td>
</tr>
<tr>
<td>Simulation: Displacement Control</td>
<td>204</td>
</tr>
<tr>
<td>Simulation: Load, Control, Factor Increment</td>
<td>205</td>
</tr>
<tr>
<td>Simulation: Dynamic Integrator, Newmark Method</td>
<td>206</td>
</tr>
<tr>
<td>Simulation: Dynamic Integrator, Hilber Hughes Taylor, HHT, $\alpha$ Method</td>
<td>207</td>
</tr>
<tr>
<td>Simulation: Absolute Convergence Criteria</td>
<td>208</td>
</tr>
<tr>
<td>Simulation: Average Convergence Criteria</td>
<td>209</td>
</tr>
<tr>
<td>Simulation: Relative Convergence Criteria</td>
<td>210</td>
</tr>
<tr>
<td>Simulation: Solution Algorithms</td>
<td>211</td>
</tr>
<tr>
<td>Simulation: Constitutive Integration Algorithm</td>
<td>212</td>
</tr>
<tr>
<td>Simulation: Status Check</td>
<td>214</td>
</tr>
<tr>
<td>Simulation: Return Value for <code>simulate</code> Command</td>
<td>215</td>
</tr>
<tr>
<td>Simulation: New Elastic Loading Case</td>
<td>225</td>
</tr>
<tr>
<td>Simulation: Combine Elastic Load Cases</td>
<td>226</td>
</tr>
<tr>
<td>1.3.6 Output Options</td>
<td>227</td>
</tr>
<tr>
<td>Output Options: Enable/Disable Output</td>
<td>228</td>
</tr>
<tr>
<td>Output Options: Enable/Disable Element Output</td>
<td>229</td>
</tr>
<tr>
<td>Output Options: Enable/Disable Displacement Output</td>
<td>230</td>
</tr>
<tr>
<td>Output Options: Enable/Disable Acceleration Output</td>
<td>231</td>
</tr>
<tr>
<td>Output Options: Enable/Disable Asynchronous Output</td>
<td>232</td>
</tr>
<tr>
<td>Output Options: Output Every n Steps</td>
<td>233</td>
</tr>
<tr>
<td>Output Options: Output Support Reactions</td>
<td>234</td>
</tr>
<tr>
<td>1.4 Checking the Model</td>
<td>235</td>
</tr>
<tr>
<td>1.5 Constitutive Testing</td>
<td>236</td>
</tr>
<tr>
<td>1.6 List of Available Commands (tentative, not up to date)</td>
<td>238</td>
</tr>
<tr>
<td>1.7 List of reserved keywords</td>
<td>248</td>
</tr>
<tr>
<td>1.8 Integrated Development Environment (IDE) for DSL</td>
<td>261</td>
</tr>
<tr>
<td>1.9 Mesh Generation using GiD</td>
<td>262</td>
</tr>
<tr>
<td>1.10 Model Development and Mesh Generation using gmesh</td>
<td>263</td>
</tr>
<tr>
<td>1.11 Model Input File Editing using Sublime</td>
<td>264</td>
</tr>
</tbody>
</table>
Chapter 1

Input, Domain Specific Language (DSL)

1.1 Chapter Summary and Highlights

1.2 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.).

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.

### 1.3 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 · 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).

#### 1.3.1 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.

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
```
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.
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 [-cfhnsmbe FILENAME]
    -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 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.

1.3.2 Finalizing Real-ESSI Program Run

To properly finalize Real ESSI program run, and close all the output files, user has to use final, closure command:

```
1 bye;
```

Command bye; has to be included at the end of each input file script, or at the end of each interactive/interpretative session, so that Real-ESSI program can gracefully exit upon execution of the script regardless of any errors which occur during execution. There are a number of alternative final commands, for example:

```
1 exit;
2 quit;
3 zdravo;
4 vozdra;
5 dvojka;
6 voljno;
7 zaijian;
```
Of course, these additional commands are equal to their pairs: zdravo ↔ здраво; vozdra ↔ воздра; dvojka ↔ двойка; voljno ↔ вољно; zaijian ↔ 再见.

### 1.3.3 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;
print var_d;
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 'whos;' produces a list of predefined variables:

Command line output
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]
  * yard = 0.9144 [m]

*- = locked variable

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:

1 L1 = 1*m; // Defines L1 to 1 m.
2 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

\[ L2 = 0.04 \ [\text{m}] \]

As additional examples, let us define few forces:

1. \( F_1 = 10 \times \text{kN}; \)
2. \( F_2 = 300 \times \text{N}; \)
3. \( F_3 = 4 \times \text{kg} \times \text{g}; \)

Here \( g \) is the predefined acceleration due to gravity.

Arithmetic operations do check (and enforce) for unit consistency. For example, \( \text{foo} = L_1 + F_1 \); produces an error because units are not compatible. However, \( \text{bar} = L_1 + L_2 \); 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).

\[ A = L_1 \times L_2; \]
\[ \text{Stress}_n = F_1 / A; \]

Units for all variables are internally converted to SI units (\( \text{kg} - \text{m} - \text{s} \)) and stored in that unit system. Variables can be displayed using different units by using the \( \text{[]} \) operator. This does not change the variable, it just displays the value of variable with required unit. For example,

\[ \begin{align*} 
1 & \quad \text{print Stress}_n; \quad \text{//Print in base SI units.} \\
2 & \quad \text{print Stress}_n \ \text{in Pa}; \quad \text{//Print in Pascal} \\
3 & \quad \text{print Stress}_n \ \text{in kPa}; \quad \text{//Print in kilo Pascal} 
\end{align*} \]

Command line output

\[ \begin{align*} 
\text{Stress}_n &= 250000 \ [\text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2}] \\
\text{Stress}_n &= 250000 \ [\text{Pa}] \\
\text{Stress}_n &= 250 \ [\text{kPa}] 
\end{align*} \]

The DSL provides functions to test the physical units of variables. For example,

\[ \text{print isForce(F1);} \]

Will print an adimensional, Boolean 1 because \( F_1 \) has units of force. While,

\[ \text{print isPressure(F);} \]

will print an adimensional, Boolean 0. The language also provides comparison of quantities with same units (remember all values are compared in SI Units).

\[ \text{print F1 > F2;} \]

will print an adimensional, Boolean 1 since \( F_1 \) is greater than \( F_2 \).

The program flow can be controlled with if and while statements, i.e.:
if (isForce(F1))
{
print F1;   // This will be executed
}

if (isForce(L1))
{
print L1;   // This will not.
}

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:

if (isForce(L1))
{
print L1;   // This will not execute
}
else
{
print L2;   // This will execute instead
}

While loops are also available:

i = 0;
while( i < 10)
{
print i;
i = i +1;
}

1.3.4 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, inch, ft]

• mass, symbol $M$, units [kg, lbm],

• time, symbol $T$, units [s]

Derived units can also be used:

• angle, symbol rad (radian), unit \([\text{dimensionless}, L/L]\]

• force, symbol N (Newton), units \([N, kN, MN, M \ast L/T^2]\),

• 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 done like this:

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.

\(^2\)See more in section 101.3.5 on page 84 of the main document
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:

```
add material  # <.> type |...|
  mass_density = <M/L^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 1.3.5. Only the material linear_elastic_isotropic_3d_LT ignores this option.

Choices for material_type are listed below
Modeling, Material Model: Linear Elastic Isotropic Material Model

The command is:

```plaintext
1 add material # <.> type linear_elastic_isotropic_3d
   mass_density = <M/L^3>
   elastic_modulus = <F/L^2>
   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 200 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: Cross Anisotropic Linear Elastic Material Model

The command is:

```
1 add material # <.> <material_number>
2   type linear_elastic_crossanisotropic
3   mass_density = <M/L^3>
4   elastic_modulus_horizontal = <F/L^2>
5   elastic_modulus_vertical = <F/L^2>
6   poisson_ratio_h_v = <.>
7   poisson_ratio_h_h = <.>
8   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 200 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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 207 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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:

```
1 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> ;
```

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 [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 207 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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:

```
1 add material # <.> type DruckerPrager
2   mass_density = <M/L^3>
3   elastic_modulus = <F/L^2>
4   poisson_ratio = <.>
5   druckerprager_k = <F/L^2>
6   kinematic_hardening_rate = <F/L^2>
7   isotropic_hardening_rate = <F/L^2>
8   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 211 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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:

```
1 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 [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 211 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederick Nonlinear Kinematic Hardening and Nonlinear 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:

```
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 \([\text{Dimensionless}]\)
- `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 211 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Kinematic Hardening

This command implements 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
1 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 211 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic
and/or Armstrong-Frederick Nonlinear Kinematic Hardening

This command implements 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:

```plaintext
1 add material # <.> type DruckerPragerNonAssociateArmstrongFrederick
   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>
   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) \([F/L^2]\)
- **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. [\]
more on this material model can be found in section 104.6.7 on page 211 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).

Figure 1.1: The physical meanings of $h_a$ and $c_r$

Figure 1.2: The physical meanings of $\xi$ and $k_d$

The physical meanings of $h_a$, $c_r$, $\xi$, and $k_d$ are shown in Figure (1.1) and Figure (1.2).
Modeling, Material Model: Rounded Mohr-Coulomb Associated Linear Isotropic Hardening Material Model

The command is:

```plaintext
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` \(\epsilon\) parameter controls roundness of the deviatoric cross-section of the yield surface. \(0.5 < \epsilon <= 1\), \(\epsilon = 0.5\) results in a triangular deviatoric section while \(\epsilon = 1\) 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 224 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: Cam Clay Material Model

The command is:

```
1  add material # <.> type CamClay
2       mass_density = <M/L^3>
3       M = <.>
4       lambda = <.>
5       kappa = <.>
6       e0 = <.>
7       p0 = <F/L^2>
8       poisson_ratio = <.>
9       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 225 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model

The command is:

```plaintext
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 radiuses of each yield surface from the smallest to the biggest. This parameter should be a string which contains the dimensionless radiuses. The radiuses 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 radiuses 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. \([F/L^2]\)
Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model that Matches $G/G_{\text{max}}$ Curves

The command is:

```
add material # <.> type vonMisesMultipleYieldSurfaceGoverGmax
  mass_density = <M/L^3>
  initial_shear_modulus = <F/L^2>
  poisson_ratio = <.>
  total_number_of_shear_modulus = <.>
  GoverGmax = <string>
  ShearStrainGamma = <string> ;
```

Command Example is

```
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";
```

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 \(G/G_{\text{max}}\) 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]
Modeling, Material Model: Drucker-Prager Nonassociated Multi-Yield Surface Material Model

The command is:

```
 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 is 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. \([F/L^2]\)
Modeling, Material Model: Drucker-Prager Nonassociated Material Model that Matches $G/G_{\text{max}}$ Curves

The command is:

```plaintext
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

```plaintext
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/Gmax 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/Gmax 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/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]
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]

- **radiuses_of_yield_surface** is the radius list of multiple yield surfaces. This parameter gives the radiiues of each yield surface from the smallest to the biggest. This parameter should be a string which contains the dimensionless radiiues. The radiiues 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 radiiues 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. \([F/L^2]\)
Modeling, Material Model: Tsinghua Liquefaction Material Model

The command is:

```plaintext
1  add material # <.> type TsinghuaLiquefactionModel
2    mass_density = <M/L^3>
3    poisson_ratio = <.>
4    initial_confining_stress = <F/L^2>
5    liquefaction_G0 = <.>
6    liquefaction_EXPN = <.>
7    liquefaction_c_h0 = <.>
8    liquefaction_mfc = <.>
9    liquefaction_mdc = <.>
10   liquefaction_dre1 = <.>
11   liquefaction_Dre2 = <.>
12   liquefaction_Dir = <.>
13   liquefaction_Alpha = <.>
14   liquefaction_gamar = <.>
15   liquefaction_pa = <.>
16   liquefaction_pmin = <.>
```

Command Example is

```plaintext
1  add material # 1 type TsinghuaLiquefactionModel
2    mass_density = 0.0* kg/m^3
3    poisson_ratio = 0.1
4    initial_confining_stress = 1E5 *Pa
5    liquefaction_G0 = 800
6    liquefaction_EXPN = 0.5
7    liquefaction_c_h0 = 1.0
8    liquefaction_mfc = 1.2
9    liquefaction_mdc = 0.4
10   liquefaction_dre1 = 0.5
11   liquefaction_Dre2 = 1500
12   liquefaction_Dir = 0.1
13   liquefaction_Alpha = 0.01
14   liquefaction_gamar = 0.01
15   liquefaction_pa = 1E5
16   liquefaction_pmin = 100 
```

where

- `mass_density` is the mass density of material \([M/L^3]\)
- `poisson_ratio` is the constant Poisson ratio \([\text{dimensionless}]\)
- `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. \([\text{dimensionless}]\)
- **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 \( G/G_{\text{max}} \) curve. When the \( G/G_{\text{max}} \) 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 \( p_{\text{min}} \) during the calculation, the pressure will be set to \( p_{\text{min}} \). This parameter can be 1. Increasing this parameter can avoid the potential numerical errors on small numbers [dimensionless].
Modeling, Material Model: SANISand Material Model, version 2004

The command is:

```
1 add material # <.> type sanisand2004
2    mass_density = <M/L^3>
3    e0 = <.>
4    sanisand2004_G0 = <.>
5    poisson_ratio = <.>
6    sanisand2004_Pat = <.>
7    sanisand2004_p_cut = <.>
8    sanisand2004_Mc = <.>
9    sanisand2004_c = <.>
10   sanisand2004_lambda_c = <.>
11   sanisand2004_xi = <.>
12   sanisand2004_ec_ref = <.>
13   sanisand2004_m = <.>
14   sanisand2004_h0 = <.>
15   sanisand2004_ch = <.>
16   sanisand2004_nb = <.>
17   sanisand2004_A0 = <.>
18   sanisand2004_nd = <.>
19   sanisand2004_z_max = <.>
20   sanisand2004_cz = <.>
21 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 \(c = M_c/Me\) \([\ ]\)
- **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 229 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: SANISand Material Model, version 2008

The command is:

```
1 add material # <.> type sanisand2008
2    mass_density = <M/L^3>
3    e0 = <.>
4    sanisand2008_G0 = <.>
5    sanisand2008_K0 = <.>
6    sanisand2008_Pat = <.>
7    sanisand2008_k_c = <.>
8    sanisand2008_alpha_cc = <.>
9    sanisand2008_c = <.>
10   sanisand2008_lambda = <.>
11   sanisand2008_ec_ref = <.>
12   sanisand2008_m = <.>
13   sanisand2008_h0 = <.>
14   sanisand2008_ch = <.>
15   sanisand2008_nb = <.>
16   sanisand2008_A0 = <.>
17   sanisand2008_nd = <.>
18   sanisand2008_p_r = <.>
19   sanisand2008_rho_c = <.>
20   sanisand2008_theta_c = <.>
21   sanisand2008_X = <.>
22   sanisand2008_z_max = <.>
23   sanisand2008_cz = <.>
24   sanisand2008_p0 = <F/L^3>
25   sanisand2008_p_in = <F/L^3>
26   algorithm = explicit (or) implicit
27   number_of_subincrements = <.>
28   maximum_number_of_iterations = <.>
29   tolerance_1 = <.>
30   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, $e_c = e_r \lambda \left(\frac{p_c}{P_{at}}\right)^x i$;

• 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 236 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Material Model: Cosserat Linear Elastic Material Model

The command is:

```plaintext
1 add material # <.> type Cosserat_linear_elastic_isotropic_3d
2   mass_density = <M/L^3>
3   lambda = <F/L^2>
4   mu = <F/L^2>
5   chi = <F/L^2>
6   pi1 = <F>
7   pi2 = <F>
8   pi3 = <F>
9 ;
```

- `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

$$
3\lambda + 2\mu + \chi \geq 0, \quad 2\mu + \chi \geq 0, \quad \chi \geq 0,
3\pi_1 + \pi_2 + \pi_3 \geq 0, \quad \pi_3 + \pi_2 \geq 0, \quad \pi_3 - \pi_2 \geq 0.
$$

(1.1)
Modeling, Material Model: von Mises Cosserat Material Model

The command is:

```
1 add material # <.> type Cosserat_von_Mises
2    mass_density = <M/L^3>
3    lambda = <F/L^2>
4    mu = <F/L^2>
5    chi = <F/L^2>
6    pi1 = <F>
7    pi2 = <F>
8    pi3 = <F>
9    plastic_internal_length = <L>
10   von_mises_radius = <F/L^2>
11   isotropic_hardening_rate = <F/L^2>
12
```

- |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).
- |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.
Modeling, Material Model: Uniaxial Linear Elastic, Fiber Material Model

The command is:

```
1 add material # <.> type uniaxial_elastic
2   elastic_modulus = <F/L^2>
3   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-present) (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.
Modeling, Material Model: Uniaxial Nonlinear Concrete, Fiber Material Model, version 02

The command is:

```plaintext
1 add material # <.> type uniaxial_concrete02
2   compressive_strength = <F/L^2>
3   strain_at_compressive_strength = <.>
4   crushing_strength = <F/L^2>
5   strain_at_crushing_strength = <.>
6   lambda = <.>
7   tensile_strength = <F/L^2>
8   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-present) (Lecture Notes URL).
Modeling, Material Model: Faria-Oliver-Cervera Concrete Material

The command is:

```plaintext
add material No (or #) <material_number>
  type FariaOliverCerveraConcrete
  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).
- `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 \([F/L^2]\)`
Modeling, Material Model: Plane Stress Layered Material

The command is:

```
1 add material No (or #) <element_number>
2    type PlaneStressLayeredMaterial
3    number_of_layers = <.>
4    thickness_array = <string>
5    thickness_scale_unit = <L>
6    with material # <string>
7 ;
```

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.
**Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 01**

The command is:

```plaintext
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-present) (Lecture Notes URL).
Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 02

The command is:

```
1 add material # <.> type uniaxial_steel02
2 yield_strength = <F/L^2>
3 elastic_modulus = <F/L^2>
4 strain_hardening_ratio = <.>
5 R0 = <.>
6 cR1 = <.>
7 cR2 = <.>
8 a1 = <.>
9 a2 = <.>
10 a3 = <>
11 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-present) (Lecture Notes URL).
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 83.

The command is:

```plaintext
1 add material No (or #) <material_number> type ↜
   PlasticDamageConcretePlaneStress
2   elastic_modulus = <F/L^2>
3   poisson_ratio = <.>
4   tensile_yield_strength = <F/L^2>
5   compressive_yield_strength = <F/L^2>
6   plastic_deformation_rate = <.>
7   damage_parameter_Ap = <.>
8   damage_parameter_An = <.>
9   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
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 NLDKGQ/Xin-Zheng-Lu, see page 83.

The command is:

```
1 add material No (or #) <material_number> type PlaneStressRebarMaterial
  with uniaxial_material # <.>
  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.
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.
Modeling, Nodes: Define Nodal Physical Group

Physical Group for nodes can be defined as well.

The command is:

```
define physical_node_group "string";
```

For example:

```
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
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
Modeling, Nodes: Removing Nodal Physical Group

Already defined node physical group `physical_node_group` can be removed.

The command is

```bash
remove physical_node_group "string";
```

For example:

```bash
remove physical_node_group "my_new_node_group";
```

this would delete the `physical_node_group"my_new_node_group"`. 
Modeling, Nodes: Print Nodal Physical Group

Printing already defined nodal physical group 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]
```
Modeling, Nodes: Removing Nodes

Nodes can be removed from the finite element model, for example during excavation, removal of finite elements.

The command is:

```plaintext
1 remove node No (or #) <.>; 
```

For example:

```plaintext
1 remove node # 1; 
```
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:

```plaintext
1 add mass to node # <.
2 mx = <M>
3 my = <M>
4 mz = <M>;
```

Similarly, the command for 6DOFs nodes (beams and shells) is:

```plaintext
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>;
```
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
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
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:

```plaintext
add elements (<.>,<.>,...) to physical_node_group "string";
```

For example:

```plaintext
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
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".
Modeling, Finite Element: Print Physical Finite Element Group

Details of the physical_element_group can be printed.

The commands is:

```
1 print physical_element_group "string";
```

For example:

```
1 print physical_element_group "my_new_element_group";
```

this would print the information about physical_element_group "my_new_element_group".

```
1 PHYSICAL_ELEMENT_GROUP my_new_element_group
2 [1 2 3]
```
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;
```
Modeling, Finite Element: Truss Element

The command is:

```
1 add element No (or #) <element_number> type truss
2     with nodes (n1, n2)
3     use material No (or #) <material_number>
4     section_area <section_area> [unit];
5     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 #)` 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 113 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Finite Element: Shear Beam Element

The command is:

```
1 add element # <.> type ShearBeam
2   with nodes (<.>, <.>)
3   cross_section = <l^2>
4   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 122 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Finite Element: Elastic Beam–Column Element

The command is:

```
1 add element # <.> type beam_elastic with nodes (<.>, <.>)
   cross_section = <L^2>
2   elastic_modulus = <F/L^2>
3   shear_modulus = <F/L^2>
4   torsion_Jx = <length^4>
5   bending_Iy = <length^4>
6   bending_Iz = <length^4>
7   mass_density = <M/L^3>
8   xz_plane_vector = (<.>, <.>, <.>)
9   joint_1_offset = (<L>, <L>, <L>)
10  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, \([F/L^3]\)
- `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, \([L]\)

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 113 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Finite Element: Timoshenko Elastic Beam–Column Element

The command is:

```
add element # <.> type beam_elastic_Timoshenko 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 = <.>
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** 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.
- **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 113 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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 \( y = 10 \text{cm}, \ z = 10 \text{cm} \) 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.
Modeling, Finite Element: Adding Fiber Section to the Finite Element Model

Fiber section can be added to the finite element model.

The command is:

```
   add section # . type FiberSection
   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 1.3.4 on page 75.

The command is:

```
   add fiber # . using material # . to section # .
   fiber_cross_section = <area>
   fiber_location = (<L>,<L>);
```

where

- \( fiber\_cross\_section \) is the area of the fiber element. (Total cross section area is the sum of all fiber areas) \( [L^2] \)

- \( fiber\_location \) location of the fiber in the beam local Y-Z plane.
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
Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element with Corotational 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).
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 116 in Lecture Notes by Jeremić et al. (1989–present) (Lecture Notes URL).
Modeling, Finite Element: 4 Node ANDES Shell with Drilling DOFs

ANDS based 3D shell element including drilling degrees of freedom. Made up by patching together 4 ANDS 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 122 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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)
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:

```plaintext
add element # <.> type 4NodeShell_NLDKGQ
  with nodes (<.>, <.>, <.>, <.>)
  section_number = <.>;
```

It can also be called using the alternative command:

```plaintext
add element # <.> type 4NodeShell_XinZhengLu_Tsinghua
  with nodes (<.>, <.>, <.>, <.>)
  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 84.
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:

```
1 add section # <.> type LayeredShellFiber
  number_of_layers = <.>
  thickness_array = "<.>,<.>..."
  with material # "<.>,<.>..."
  thickness_scale_unit = <L>
  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 54 and 55.
- `thickness_scale_unit` is the total thickness of the section
- `outofplane_shear_modulus` is the out-of-plane shear modulus of the section
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.
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:

```plaintext
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 #) <material_number>` 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.
Modeling, Finite Element: SuperElementLinearElasticImport

The command is:

```plaintext
add element No (or #) <element_number>
  type SuperElementLinearElasticImport
  with hdf5_file = <string>
```

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 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).
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

![8 Node Brick Element Diagram](image_url)

- 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 101 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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 x 3 x 3), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction (3 x 3 x 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 102.4.2 on page 103 in Lecture Notes by Jeremić et al.
(1989-present) (Lecture Notes URL).
Modeling, Finite Element: 27 Node Brick Element

The command is:

```
1 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

```
1 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 102.4.3 on page 105 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Finite Element: Variable 8-27 Node Brick Element

The command is:

```plaintext
add element No (or #) <element_number> type variable_node_brick_8_to_27
   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 No (or #) <material_number>;
```

and/or

```plaintext
add element No (or #) <element_number> type variable_node_brick_8_to_27
   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 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
Modeling, Finite Element: 8 Node Brick u-p Element

The command is:

```plaintext
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^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_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) 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 ($n = \frac{V_{\text{voids}}}{V_{\text{total}}}$) 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-present) (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-present) (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 $k = K/\rho_f g$, 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 on page 143 in Lecture Notes by Jeremić et al. (1989-present) (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-present) (Lecture Notes URL).
Modeling, Finite Element: 20 Node Brick u-p Element

The command is:

```plaintext
1 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
1 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 ($n = V_{\text{voids}}/V_{\text{total}}$) 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-present) (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-present) (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 on page 143 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Finite Element: 27 Node Brick u-p Element

The command is:

```plaintext
add element # <.> type 27NodeBrick_up
   using <.> Gauss points each direction
```
and/or

```plaintext
add element # <.> type 27NodeBrick_up
```
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 (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-present) (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-present) (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. $[length]/[time]$. 

Real-ESI Domain Specific Language  
version: March 30, 2020, 14:01
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 on page 143 of the main document; Description of output by this element can be found in section 206.8.2.
Modeling, Finite Element: 8 Node Brick u-p-U Element

The command is:

```plaintext
1 add element # <.> type 8NodeBrick_upU 
2     using <.> Gauss points each direction 
3     with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) 
4     use material # <.> 
5     porosity = <.> 
6     alpha = <.> 
7     rho_s = <M/L^2> 
8     rho_f = <M/L^2> 
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_upU 
2     with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) 
3     use material # <.> 
4     porosity = <.> 
5     alpha = <.> 
6     rho_s = <M/L^2> 
7     rho_f = <M/L^2> 
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-present) (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-present) (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.
More on theory for this finite element can be found in section 102.12.1 on page 139 in Lecture Notes by Jeremić et al. (1989-present) (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-present) (Lecture Notes URL).
Modeling, Finite Element: 20 Node Brick u-p-U Element

The command is:

```
1 add element # <.> type 20NodeBrick_upU
2   using <.> Gauss points each direction
3   with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
4   use material # <.>
5   porosity = <.>
6   alpha = <.>
7   rho_s = <M/L^2>
8   rho_f = <M/L^2>
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

```
1 add element # <.> type 20NodeBrick_upU
2   with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>)
3   use material # <.>
4   porosity = <.>
5   alpha = <.>
6   rho_s = <M/L^2>
7   rho_f = <M/L^2>
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, 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 \(n = V_{\text{voids}}/V_{\text{total}}\) 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-present) (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-present) (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.
More on theory for this finite element can be found in section 102.12.1 on page 139 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL). Description of output by this element can be found in section 206.8.2.
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 (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 x 3 x 3), while for 20 nodes, 8-20 node and 8-27 node bricks 3 Gauss points are used in each direction (3 x 3 x 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 \( n = \frac{V_{\text{voids}}}{V_{\text{total}}} \) 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-present) (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-present) (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 102.12.1 on page 139 of the main document; Description of output by this element can be found in section 206.8.2.
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.
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.
Modeling, Finite Element: Force Based Dry Hard Contact/Interface/Joint Element

The command is:

```plaintext
1 add element # <.> type ForceBasedHardContact
2   with nodes (<.>, <.>)
3     axial_stiffness = <F/L>
4     shear_stiffness = <F/L>
5     axial_viscous_damping = <F/L>
6     shear_viscous_damping = <F/L>
7     friction_ratio = <.>
8     contact_plane_vector = (<.>, <.>, <.>);
```

The axial force $F_a$ and axial stiffness $E_a$ in defined as

$$F_a = E_a \times \delta_a$$

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.

**REALLY IMPORTANT NOTE:** `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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Force Based Dry Soft Contact/Interface/Joint Element

The command is:

```plaintext
1 add element # <.> type ForceBasedSoftContact
2     with nodes (<.>, <.>)
3     initial_axial_stiffness = <F/L>
4     stiffening_rate = <1/m>
5     max_axial_stiffness = <F/L>
6     shear_stiffness = <F/L>
7     axial_viscous_damping = <F/L>
8     shear_viscous_damping = <F/L>
9     friction_ratio = <.>
10    contact_plane_vector = (<.>, <.>, <.>);
```

The axial force $F_a$ and axial stiffness $E_a$ in defined as

$$
F_a = b \ast \exp(a \ast \delta_a) \ast \delta_a
$$
$$
E_a = \max(b \ast \exp(a \ast \delta_a) \ast (1 + a \ast \delta_a), E_{max})
$$

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).
- `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.

**REALLY IMPORTANT NOTE:** `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 have convergence issues.
Description of output by this element can be found in section 206.8.7
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 $E_a$ in defined as

\[ F_a = b \cdot \delta_a \]
\[ E_a = b \]

(1.4)

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.

**REALY IMPORTANT NOTE:** 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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Force Based Coupled Soft Contact/Interface/Joint Element

The command is:

```plaintext
1 add element # <.> type ForceBasedCoupledSoftContact
   with nodes (<.>, <.>)
   initial_axial_stiffness = <F/L>
   stiffening_rate = <1/m>
   max_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

\[
F_a = b \cdot \exp(a \cdot \delta_a) \cdot \delta_a \\
E_a = \max(b \cdot \exp(a \cdot \delta_a) \cdot (1 + a \cdot \delta_a), E_{max})
\]  

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 \( \exp(a \cdot \delta_a) \) 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 \( u_p \) 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 the Coulomb friction ratio.
- `contact_plane_vector` defines the normal to the contact/interface/joint plane.
**REALLY IMPORTANT NOTE:** `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 have convergence issues.

Description of output by this element can be found in section 206.8.7.
Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior

The command is:

```plaintext
1 add element # <.> type StressBasedHardContact_ElPPlShear
2   with nodes (<.>, <.>)
3   axial_stiffness = <Pa>
4   initial_shear_stiffness = <Pa>
5   axial_viscous_damping = <Pa*s>
6   shear_viscous_damping = <Pa*s>
7   residual_friction_coefficient = <.>
8   shear_zone_thickness = <m>
9   contact_plane_vector = (<.>, <.>, <.> );
```

and/or:

```plaintext
1 add element # <.> type StressBasedHardContact_ElPPlShear
2   with nodes (<.>, <.>)
3   axial_stiffness = <Pa>
4   initial_shear_stiffness = <Pa>
5   axial_viscous_damping = <Pa*s>
6   shear_viscous_damping = <Pa*s>
7   residual_friction_coefficient = <.>
8   shear_zone_thickness = <m>
9   surface_vector_relative_tolerance = <.>;
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$\sigma_a = b \epsilon_a$$

$$E_a = b$$  \hspace{1cm} (1.6)

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 101$kPa$ axial stress described in Section 104.7.3
- **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 friction coefficient described in Section 104.7.3.

• **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.

**REALLY IMPORTANT NOTE:** **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 have convergence issues.

Description of output by this element can be found in section 206.8.7
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

$$
\sigma_a = b \ast \epsilon_a
$$

$$
E_a = b
$$

where

$$
E_a = b \text{ refers to the axial stiffness in axial contact/interface/joint direction,}
$$

$\epsilon_a$ 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 
- `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

- **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.

**REALLY IMPORTANT NOTE:** 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 have convergence issues.

Description of output by this element can be found in section 206.8.7
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>
  contact_plane_vector = (<.>, <.>, <.>);
```

The axial stress \( \sigma_a \) and axial stiffness \( E_a \) is defined as

\[
\sigma_a = b \ast \epsilon_a \\
E_a = b
\]

(1.8)

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

- **rate_of_softening** \(R_s\) is the parameter to control the rate of frictional softening described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power \(n\) with incremental form as

\[
\Delta \mu = -\frac{n * R_s (\mu_p - \mu_r)}{(\pi/2)^n \theta^{1/n-1}} * \cos^2 \theta \Delta \gamma^p
\]

(1.9)

\[
\theta = \frac{\mu_p - \mu}{\mu_p - \mu_r} (\pi/2)^n
\]

(1.10)

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.

\[
\Delta \gamma^p = \sqrt{\Delta \epsilon^p_{ij} \Delta \epsilon^p_{ij}}
\]

(1.11)

- **size_of_peak_plateau** \(n\) is the frictional softening parameter to control the size of plateau as described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power \(n\) with incremental form as shown in Equation 1.31.

- **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

\[
\mu_p = \max(\mu_p0, \mu_p0 - k * \log(\sigma_a/P_0))
\]

(1.12)

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

- **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.
REALLY IMPORTANT NOTE: 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 have convergence issues.

Description of output by this element can be found in section 206.8.7
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;
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 $E_a$ are defined as

\[
\sigma_a = b \exp(a \epsilon_a) \epsilon_a
\]
\[
E_a = \max\left(b \exp(a \epsilon_a) \left(1 + a \epsilon_a \right), E_{\max}\right)
\]

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).
• **stiffening_rate** (a) Represents exponential stiffening rate $e^{a \epsilon_a}$ 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

• **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

• **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.

**REALLY IMPORTANT NOTE:** 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 have convergence issues.

Description of output by this element can be found in section 206.8.7
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

$$
\sigma_a = b \times \exp(a \times \epsilon_a) \times \epsilon_a
$$

$$
E_a = \max(b \times \exp(a \times \epsilon_a) \times (1 + a \times \epsilon_a), E_{max})
$$

where

- $b$ refers to the initial axial stiffness in axial contact/interface/joint direction,
- $a$ refers to the stiffening rate in axial contact direction,
- $E_{max}$ 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 1 m
penetration.

- **stiffening_rate (a)** Represents exponential stiffening rate \( \exp(sr \times \epsilon_a) \) 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 101\( kPa \) axial stress described in Section 104.7.3

- **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

- **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.

**REALLY IMPORTANT NOTE:** 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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior

The command is:

```plaintext
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_plane_vector = (<.>, <.>, <.>);
```

and/or;

```plaintext
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 $E_a$ in defined as

$$
\sigma_a = b \ast \exp(a \ast \epsilon_a) \ast \epsilon_a
$$

$$
E_a = \max(b \ast \exp(a \ast \epsilon_a) \ast (1 + a \ast \epsilon_a), E_{\text{max}})
$$

(1.15)

where

$b$ refers to the initial axial stiffness in axial contact/interface/joint direction,

$a$ refers to the stiffening rate in axial contact direction,

$E_{\text{max}}$ 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 $\exp(sr * \epsilon_n)$ 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
- **rate_of_softening** ($R_s$) Is the parameter to control the rate of frictional softening described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as
  \[
  \Delta\mu = -\frac{R_s(\mu_p - \mu_r)}{(\pi/2)^n \theta^{1/n-1}} \cos^2\theta \Delta\gamma^p
  \]
  \[
  \theta = \frac{\mu_p - \mu}{\mu_p - \mu_r}(\pi/2)^n
  \]
  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.
  \[
  \Delta\gamma^p = \sqrt{\Delta\epsilon^p_{ij}\Delta\epsilon^p_{ij}}
  \]
- **size_of_peak_plateau** ($n$) Is the frictional softening parameter to control the size of plateau as described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as shown in Equation 1.31.
- **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
  \[
  \mu_p = \max(\mu_{p0}, \mu_{p0} - k \times \log(\sigma_a/P_0))
  \]
where \( \mu_\text{p0} \) is the peak frictional hardening limit, \( k \) is the peak frictional parameter rate of decrease and \( P_0 \) is the reference stress of \( P_0 = 101kPa \).

- \text{residual\_friction\_coefficient}(\mu_r) \) is the residual frictional parameter as described in Section 104.7.3
- \text{shear\_zone\_thickness} \( h \) is the shear zone thickness
- \text{contact\_plane\_vector} Vector defining the normal to the contact\_interface\_joint plane.
- \text{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.

\textbf{REALLY IMPORTANT NOTE:} \text{contact\_plane\_vector} defines a direction from Node I to Node J, that is, \textbf{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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior

The command is:

```plaintext
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

$$\sigma_a = b * \epsilon_a$$

$$E_a = b$$

(1.20)

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 101 kPa axial stress described in Section 104.7.3

• *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 friction coefficient described in Section 104.7.3

• *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.

**REALLY IMPORTANT NOTE:** *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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Non-linear Hardening Shear Behavior

The command is:

```plaintext
add element # <.> type StressBasedCoupledHardContact_NonLinHardShear
  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;

```plaintext
add element # <.> type StressBasedCoupledHardContact_NonLinHardShear
  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 $E_a$ are defined as

$$\sigma_a = b \ast \epsilon_a$$

$$E_a = b$$

(1.21)

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 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

• 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

• 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.

**REALLY IMPORTANT NOTE:** 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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Non-linear Hardening and Softening Shear Behavior

The command is:

```plaintext
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_zone_thickness = (.<>, <.>, <.> );
```

The axial stress $\sigma_a$ and axial stiffness $E_a$ in defined as

$$\sigma_a = b \times \epsilon_a$$

$$E_a = b$$

(1.22)

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 1m penetration.

• axial_penalty_stiffness (Ep) 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 (Es) Is the stiffness in the tangential (shear, local y or z axis) directions at 101kPa axial stress, described in Section 104.7.3.

• rate_of_softening (Rs) Is the parameter to control the rate of frictional softening described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power n with incremental form as

\[
\Delta \mu = -\frac{n \cdot R_s (\mu_p - \mu_r)}{\left(\frac{\pi}{2}\right)^n \theta^{1/n - 1}} \cos^2 \theta \Delta \gamma^p
\]

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.

\[
\Delta \gamma^p = \sqrt{\Delta \epsilon_{ij}^p \Delta \epsilon_{ij}^p}
\]

• size_of_peak_plateau (n) Is the frictional softening parameter to control the size of plateau as described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power n with incremental form as shown in Equation 1.31.

• 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

\[
\mu_p = \max(\mu_p0, \mu_p0 - k \cdot \log(\sigma_a/P_0))
\]

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 \( P_0 = 101kPa \).

• residual_friction_coefficient (\( \mu_r \)) Is the residual frictional parameter as described in Section 104.7.3.

• 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.

**REALLY IMPORTANT NOTE**: *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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior

The command is:

```plaintext
add element # <.> type StressBasedCoupledSoftContact_ElPPlShear
  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:

```plaintext
add element # <.> type StressBasedCoupledSoftContact_ElPPlShear
  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

$$\sigma_a = b \cdot \exp(a \cdot \epsilon_a) \cdot \epsilon_a$$

$$E_a = \max(b \cdot \exp(a \cdot \epsilon_a) \cdot (1 + a \cdot \epsilon_a), E_{max})$$

(1.27)

where

- $b$ refers to the initial axial stiffness in axial contact/interface/joint direction,
- $a$ refers to the stiffening rate in axial contact direction,
- $E_{max}$ 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 \( \exp(sr \times \epsilon_n) \) 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 \( u_p \) 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 101kPa axial stress described in Section 104.7.3

• `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 friction coefficient described in Section 104.7.3

• `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.

**REALLY IMPORTANT NOTE:** `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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Non-linear Hardening Shear Behavior

The command is:

```
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>
  contact_plane_vector = (<.>, <.>, <.>);
```

and/or:

```
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

$$
\sigma_a = b \times \exp(a \times \epsilon_a) \times \epsilon_a \\
E_a = \max(b \times \exp(a \times \epsilon_a) \times (1 + a \times \epsilon_a), E_{max})
$$

(1.28)

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 $\exp(sr \times \epsilon_n)$ 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 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

- **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

- **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.

**REALLY IMPORTANT NOTE:** **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 have convergence issues.

Description of output by this element can be found in section 206.8.7
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Non-linear Hardening and Softening Shear Behavior

The command is:

```
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:

```
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 $E_a$ are defined as

$$
\sigma_a = b \cdot e^{a \cdot \epsilon_a} \cdot \epsilon_a
$$

$$
E_a = \max(b \cdot e^{a \cdot \epsilon_a} \cdot (1 + a \cdot \epsilon_a), E_{\max})
$$

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 $exp(sr * \epsilon_a)$ 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
- **rate_of_softening** ($R_s$) Is the parameter to control the rate of frictional softening described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as

$$
\Delta \mu = -\frac{n * R_s}{(\pi/2)^n} \left(\frac{\mu_p - \mu_r}{\pi/2}\right)^n \cos^2 \theta \Delta \gamma^p
$$

(1.30)

$$
\theta = \frac{\mu_p - \mu}{\mu_p - \mu_r} (\pi/2)^n
$$

(1.31)

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.

$$
\Delta \gamma^p = \sqrt{\Delta \epsilon_{ij}^p \Delta \epsilon_{ij}^p}
$$

(1.32)

- **size_of_peak_plateau** ($n$) Is the frictional softening parameter to control the size of plateau as described in Section 104.7.3. The frictional softening function is an inverse tangent function raised to power $n$ with incremental form as shown in Equation 1.31.
- **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

$$\mu_p = \max(\mu_{p0}, \mu_{p0} - k \cdot \log(\sigma_a/P_0))$$ (1.33)

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

• 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.

**REALLY IMPORTANT NOTE**: 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 have convergence issues.

Description of output by this element can be found in section 206.8.7
**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 123 in Lecture Notes by Jeremić et al. (1989-present) (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 123 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Finite Element: Frictional Pendulum Isolator/Dissipator Finite Element version01

(command syntax is in development),

more on this finite element can be found in section 102.11 on page 123 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
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 123 in Lecture Notes by Jeremić et al. (1989-present) (Lecture Notes URL).
Modeling, Damping: Adding Rayleigh Damping

First, define the Rayleigh damping.

```plaintext
1 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
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.
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.
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.
Modeling, Constraints and Supports: Adding Constraints or Supports

1 \texttt{fix node \# <.> dofs [ux uy ux p Ux Uy Uz rx ry rz all]};

where at least one of the DOF fixity codes (ux uy ux p Ux Uy Uz rx ry rz all) has to be invoked. These codes are

- \texttt{ux}, translation in \textit{x} direction, for structures and solids (solid phase only in \textit{u}−\textit{p}−\textit{U} and \textit{u}−\textit{p} elements)
- \texttt{uy}, translation in \textit{y} direction, for structures and solids (solid phase only in \textit{u}−\textit{p}−\textit{U} and \textit{u}−\textit{p} elements)
- \texttt{uz}, translation in \textit{z} direction, for structures and solids (solid phase only in \textit{u}−\textit{p}−\textit{U} and \textit{u}−\textit{p} elements)
- \texttt{p}, pore fluid pressure (for fluid phase in \textit{u}−\textit{p}−\textit{U} and \textit{u}−\textit{p} elements)
- \texttt{Ux}, translation of pore fluid phase in \textit{x} direction (for \textit{u}−\textit{p}−\textit{U} elements)
- \texttt{Uy}, translation of pore fluid phase in \textit{y} direction (for \textit{u}−\textit{p}−\textit{U} elements)
- \texttt{Uz}, translation of pore fluid phase in \textit{z} direction (for \textit{u}−\textit{p}−\textit{U} elements)
- \texttt{rx}, rotation around \textit{x} axes (for structural elements)
- \texttt{ry}, rotation around \textit{y} axes (for structural elements)
- \texttt{rz}, rotation around \textit{z} axes (for structural elements)
- \texttt{all}, all applicable DOFs for a given node

Example fix translation \textit{x} and \textit{y} for node \#3 \texttt{fix node \# 3 dofs ux uy};
Example fix all appropriate DOFs for node \#7. \texttt{fix node \# 7 dofs all};
**Modeling, Constraints and Supports: Free Constraint or Support**

Free the specified DOFs on a designated node.

```
1 free node # <.> dofs [ux uy ux p Ux Uy Uz rx ry rz];
```
Modeling, Constraints and Supports: Add Master-Slave Nodes for the Same DOFs

Add the equal dof for master-slave nodes for the same degree of freedom.

```plaintext
1 add constraint equal_dof with 
2 master node # <.> and 
3 slave node # <.> 
4 dof to constrain <.>; 
```
Modeling, Constraints and Supports: Adding Master-Slave Nodes for Different DOFs

Add the equal dof for master-slave nodes for different degree of freedom.

```
1 add constraint equal_dof with node # .> dof .> master and node # <- .> dof .> slave;
```
Modeling, Constraints and Supports: Remove Master-Slave equal DOFs

Remove the master-slave equal_dofs.

```plaintext
1  remove constraint equal_dof node # <.>
```
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 <.>`
Modeling, Acceleration Field: Adding Acceleration/Inertia Field

```
1 add acceleration field # <.
2   ax = <acceleration in x direction>*[L/T^2]
3   ay = <acceleration in y direction>*[L/T^2]
4   az = <acceleration in z direction>*[L/T^2];
```

Example adding acceleration induced loading field for (some) elements

```
1 add acceleration field # 1
2   ax = 0*m/s^2
3   ay = 0*m/s^2
4   az = -9.81*m/s^2;
```

**NOTE:** see note on page 169 for command

```
1 add load # <.> to element # <.> type self_weight use acceleration ← field # <.>
```
Modeling, Loads: Nodal Loads

The general signature to add loads is

```plaintext
add load # <.> to node # <.>
  type <load type> <direction> = <force_amplitude>
{ 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
- **Fy**, force in \( y \) direction
- **Fz**, force in \( z \) direction
- **F_fluid_x**, force to the pore fluid phase in \( x \) direction
- **F_fluid_y**, force to the pore fluid phase in \( y \) direction
- **F_fluid_z**, force to the pore fluid phase in \( z \) direction
- **Mx**, moment about \( x \) axes
- **My**, moment about \( y \) axes
- **Mz**, moment about \( z \) axes

Example command for adding three linear forces \((f_x = -10 \ast kN, f_y = -10 \ast kN, f_z = -10 \ast kN)\) to node # 1:

---

1. Applies to solid phase only when connected to coupled elements
2. Applies to fluid phase when connected to coupled elements
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.
Modeling, Loads: Nodal Path Loads

To add forces which follow a path other than linear, we have the \texttt{path\_series}, for equally spaced time series data, and \texttt{path\_time\_series} for variable spaced time series data.

The commands are:

```plaintext
1 add load # <.> to node # <.> type path\_series
2 FORCETYPE = <force or moment scale factor>
3 time\_step = <T>
4 series\_file = "STRING";
```

```plaintext
1 add load # <.> to node # <.> type path\_time\_series
2 FORCETYPE = <force or moment scale factor>
3 series\_file = "STRING";
```

As before, \texttt{FORCETYPE} can be $F_x$, $F_y$, $F_z$, $M_x$, $M_y$, $M_z$, $F_{\text{fluidx}}$, $F_{\text{fluidy}}$, $F_{\text{fluidz}}$.

The format of the series\_file is one column of text for the equally spaced case (\texttt{path\_series}) and double column, one for time and second one for data values, for (\texttt{path\_time\_series}).
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.
Modeling, Loads: Selfweight Element Load

```
add load # <.

to element # <.

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 $0.0981 \text{m/s}^2$ and then apply it in 100 seconds. This is to be done in command add acceleration field ... on page 164.
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>;
```
Modeling, Loads: 8 Node Brick Surface Load with Variable Pressure

Surface of 8 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> );
```
Modeling, Loads: 20Node Brick Surface Load with the Constant Pressure

Surface of 20 node brick element with same pressure magnitudes at all nodes:

```plaintext
add load # <.> to element # <.> type surface at nodes (<.> , <.> , <.> , <.> , <.> , <.> , <.> , <.> , <.> , <.> ) with magnitude <Pa>;
```
Modeling, Loads: 20 Node Brick, Surface Load with Variable Pressure

Surface of 20 node brick element with variable pressure magnitudes at all nodes:

```plaintext
```
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
add load # <.> to element # <.> type surface at nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>) with magnitude <Pa>;
```
Modeling, Loads: 27 Node Brick Surface Load with Variable Pressure

Surface of 27 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>, <Pa>, <Pa>, <Pa>, <Pa>);
```
Modeling, Loads: Removing Loads

Loads can be removed using:

```plaintext
remove load # <..>
```
Modeling, Loads: Domain Reduction Method, DRM

```plaintext
1 add load # . type domain reduction method hdf5_file = <string>;
2 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.(1.3), eight components are required for the DRM input.

![DRM_input_hdf5](image.png)

Figure 1.3: 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
1 # Created by Jose Antonio Abell Mena
2 # This file reads old-format DRM input files and translates them into new
3 # HDF5-based format.
4 #
5 # This file produces a rigid body input to the DRM layer. That is, all DRM
6 # nodes have same X-direction displacement and acceleration. In this case a
7 # sine wave is used. This is not realistic, it's just for demonstration
8 # purposes. DRM won't work in this case but can be used to verify input if a
9 # pseudo-static analysis is done (zero density on all elements and apply loads
10 # with transient analysis.)
11 # For real input motions, produced using some other means, say SW4, fk,
12 # Hisada or MS ESSI program, this simple program can be used as an example,
13 # where DRM input file format used here can be (re)used, while those real
14 # motions are read from 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.drminput","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" → 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))

# 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

# Format is:
# Array/matrix for Accelerations and Displacements has the following → shape
# [3*(N_boundary_nodes + N_exterior_nodes) , Ntimesteps]
# where component

# A[3*n], A[3*n+1], A[3*n+2]
# correspond to accelerations/displacements in X, Y, and Z directions
# at node n.
# The location corresponding to node n is that of the n-th component of array
# "DRM Nodes"

# 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()
Modeling, Wave Field for Creating DRM Loads: Add Wave Field

```plaintext
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

```plaintext
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.
<table>
<thead>
<tr>
<th></th>
<th>Vs</th>
<th>Vp</th>
<th>rho</th>
<th>damp</th>
<th>thickness</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>200</td>
<td>333</td>
<td>2000</td>
<td>0.03</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>408</td>
<td>2000</td>
<td>0.04</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>2000</td>
<td>3400</td>
<td>2400</td>
<td>0.05</td>
<td></td>
</tr>
</tbody>
</table>
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.

```plaintext
generate wave propagation results of wave field
# <.> at depth <L> to <output_filename_prefix>
```

One example of deconvolution is

```plaintext
generate wave propagation results of wave field
# 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`.
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.

```
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

```
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> ;
```

```
1 generate DRM motion file from wave field
2 # <.> in direction <ux|uy|uz>
3 # <.> in direction <ux|uy|uz>
4 # <.> in direction <ux|uy|uz>
5 soil_surface at z = <L>
6 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.
Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident SV Wave

```plaintext
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 adding an inclined plane wave field

```plaintext
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
  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 1.4, 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_incident_angle** specifies the inclination angle of incident SV wave, measured from vertical axis of wave plane to the wave propagation axis. In figure 1.4, the incident angle of SV wave is $\theta$.

- **motion_time_step** is the time step/interval used for discretizing the harmonic motion into time domain.

- **number_of_time_steps** is the number of total time steps for the discretized harmonic motion.

- **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.

<table>
<thead>
<tr>
<th></th>
<th>Vs</th>
<th>Vp</th>
<th>rho</th>
<th>damp</th>
<th>thickness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>333</td>
<td>2000</td>
<td>0.02</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>408</td>
<td>2000</td>
<td>0.02</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>2000</td>
<td>3400</td>
<td>2400</td>
<td>0.02</td>
<td></td>
</tr>
</tbody>
</table>

- **soil_surface at <x|y|z>** defines the location of soil surface in the global coordinate system of Real-ESSI.
Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident P Wave

```plaintext
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

```plaintext
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 1.4, 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 1.4, 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.

<table>
<thead>
<tr>
<th></th>
<th>Vs</th>
<th>Vp</th>
<th>rho</th>
<th>damp</th>
<th>thickness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>//</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>333</td>
<td>2000</td>
<td>0.02</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>408</td>
<td>2000</td>
<td>0.02</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>2000</td>
<td>3400</td>
<td>2400</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( \text{soil\_surface\ at \ <x|y|z>} \) defines the location of soil surface in the global coordinate system of Real-ESSI.
Modeling, Wave Field for Creating DRM Loads: DRM Inclined Motion

This command generates inclined DRM motion with pre-defined inclined wave field.

```
1 generate DRM motion file from wave field # <.> hdf5_file = <string>;
```

One example of generating inclined DRM motion is:

```
1 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.
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.

```plaintext
1 add load # <.> type imposed motion to node # <.> dof DOFTYPE
2 time_step = <T>
3 displacement_scale_unit = <L>
4 displacement_file = "filename"
5 velocity_scale_unit = <L/T>
6 velocity_file = "filename"
7 acceleration_scale_unit = <L/L^2>
8 acceleration_file = "filename";
```

The above command generates load to the corresponding node to get the applied imposed motion.
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 is 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.
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:

```
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.
Modeling, Imposed Motions: Remove Imposed Motions

Motions can be removed using:

```plaintext
remove imposed motion # <.>
```
1.3.5 Simulation
Simulation, Solvers: Sequential Solvers

```plaintext
1 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.
Simulation, Solvers: Parallel Solvers

```c
define solver parallel petsc <petsc_options> ;
```

Direct Solvers

Command Example for a direct solver:

```c
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:

```c
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 1.1 and 1.2 on next pages present a full set of options for iterative solvers and preconditioners.
<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>
<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>
## Simulation: Static Solution Advancement

1. `simulate <.> steps using static algorithm;`
Simulation: Dynamic Solution Advancement with the Constant Time Step

```plaintext
simulate <.
steps using transient algorithm time_step = <T>;
```
Simulation: Dynamic Solution Advancement with Variable Time Step

1. simulate < . > steps using variable transient algorithm
2.     time_step = < T >
3.     minimum_time_step = < . >
4.     maximum_time_step = < . >
5.     number_of_iterations = < . >
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:

```plaintext
1 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.
Simulation: Displacement Control

```plaintext
define static integrator displacement_control using node # <.> dof ←
  DOFTYPE increment <L>;
```
Simulation: Load, Control, Factor Increment

```plaintext
1 define load factor increment <.>;
```
Simulation: Dynamic Integrator, Newmark Method

```plaintext
1 define dynamic integrator Newmark with gamma = <.> beta = <.>;
```
Simulation: Dynamic Integrator, Hilber Hughes Taylor, HHT, $\alpha$ Method

1 define dynamic integrator Hilber_Hughes_Taylor with alpha = <.> ;
Simulation: Absolute Convergence Criteria

```plaintext
1 define convergence test
2 < Absolute_Norm_Unbalanced_Force | ←
   Absolute_Norm_Displacement_Increment >
3 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/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.
Simulation: Average Convergence Criteria

```plaintext
define convergence test ←
   <Average_Norm_Unbalanced_Force | Average_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 (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.
**Simulation: Relative Convergence Criteria**

```plaintext
define convergence test ←
   <Relative_Norm_Unbalanced_Force | Relative_Norm_Displacement_Increment>

tolerance = <.>
minimum_absolute_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 (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.
Simulation: Solution Algorithms

1 define algorithm < With_no_convergence_check | linear_elastic | ←
   Newton | Modified_Newton | Newton_With_LineSearch >;

1 define algorithm < Newton_With_Subincrement >
2   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).

- $\text{minimum\_time\_step}$ 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 $\text{minimum\_time\_step}$, the algorithm returns failure to convergence.

Note: If any Newton algorithm is used, the convergence test should be defined before the algorithms.
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:

```plaintext
1 define NDMaterial constitutive integration algorithm Forward_Euler;
```

```plaintext
1 define NDMaterial constitutive integration algorithm ← Forward_Euler_Subincrement
2 number_of_subincrements = <.> ;
```

```plaintext
1 define NDMaterial constitutive integration algorithm Backward_Euler
2 yield_function_relative_tolerance = <.>
3 stress_relative_tolerance = <.>
4 maximum_iterations = <.>;
```

```plaintext
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.
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_Pat = atmospheric_pressure
sanisand2004_p_cut = 0.1 * atmospheric_pressure
sanisand2004_c = 0.712
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_z_max = 4.
sanisand2004_A0 = 0.704
sanisand2004_nd = 3.5
sanisand2004_xi = 0.7
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 1.3.5). Therefore, the second branch of the ‘if’ statement will execute.
**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 1.5, 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.

```plaintext
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 1.5: 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, 1991) 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
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;
    } 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;
    }
}

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

Checking model......................... Pass !
Checking constraint ................. Pass !
Checking numberer..................... Pass !
Checking algorithm.................... Pass !
Checking system of equation ........ Pass !
Checking static integration .......... 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.................................................................
Checking constraint ←
  handler...............................................................
handler ................................................................................................. Pass!
69 Checking numberer .............................................................................. Pass!
70 Checking analysis .............................................................................. Pass!
71 Checking system of equation .............................................................. Pass!
72 Checking static integration .................................................................. Pass!
73
74 Static Analysis: [1 /1 ]
75 [iteration 1 /5 ] Convergence Test: Absolute Norm: (tol: 0.001)
    Displacement Increment:
    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
82 [iteration 2 /5 ] Convergence Test: Absolute Norm: (tol: 0.001)
    Displacement Increment:
    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
89 > Analysis End

90
91 Starting sequential static multi-step analysis
92 ====================================================================================================
94 Creating analysis .............................................................................. Pass!
95 Checking constraint .............................................................................. Pass!
96 Checking numberer .............................................................................. Pass!
97 Checking analysis .............................................................................. Pass!
98 Checking system of equation .............................................................. Pass!
99 Checking static integration .................................................................. Pass!
100
101 Static Analysis: [1 /1 ]
102 [iteration 1 /5 ] Convergence Test: Absolute Norm: (tol: 0.001)
    Displacement Increment:
    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: 0.39156
Average Norm deltaU: 0.0006265
Relative Norm deltaF: 0.39156
Relative Norm deltaU: 0.39156

[iteration 4 /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 5 /5 ] Convergence Test: Absolute Norm ←
Displacement 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 ←
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
Checking constraint ⇐
handler
Checking ⇐
numberer
Checking analysis ⇐
algorithm
Checking system of equation ⇐
handler
Checking static integration ⇐
handler

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
Checking constraint ⇐
handler
Checking ⇐
numberer
Checking analysis ⇐
algorithm
Checking system of equation ⇐
handler
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 ←

Real-ESSI Domain Specific Language
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-05
Relative Norm deltaU: 2.1658e-19

> Analysis End ←

-------------------------------------------------------------------------------

How polite! Bye, have a nice day!
Simulation: New Elastic Loading Case

For design applications, linear elastic analysis cases are performed and later combined, using factors of safety (see section 1.3.5 on page 226) 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.
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`. 
1.3.6 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.
**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

```
1 disable all output;
```

Command to enable output is

```
1 enable all output;
```
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;
```
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

```plaintext
1 disable displacement output;
```

Command to enable displacement output is

```plaintext
1 enable displacement output;
```
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;
```
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

```1 disable asynchronous output;```

Command to enable asynchronous output is

```enable asynchronous output;```
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

```
output every <> steps;
```

For example: To output only at interval of two time steps for a simulation of 100 steps. One can write

```
output every 2 steps;
```

This will only output for steps 2,4,6,... until 100th step.
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;
```
1.4 Checking the Model

Real-ESSI provides model check capability:

```bash
check model;
```

This will iterate over domain components (Nodes, Elements, Loads, Constraints, etc.) and execute the `checkModel()` function. Each domain component writes to the terminal if it has encountered an error which gets recorded in `essi.log` file as well. For example, bricks will report when the computed Jacobian is negative and other types of error.

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.

Besides, this command will check if any defined node is never used. If any defined node is never used, `essi` will report the node number and exit.

This command can be thought as a dry run that is used to check the model before a full analysis.
1.5 Constitutive Testing

Material models can be tested using constitutive drivers which exercise single material models. RealESSI 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:

Stress.feioutput → $\sigma_{11}$ $\sigma_{22}$ $\sigma_{33}$ $\sigma_{12}$ $\sigma_{13}$ $\sigma_{23}$.

Strain.feioutput → $\epsilon_{11}$ $\epsilon_{22}$ $\epsilon_{33}$ $\epsilon_{12}$ $\epsilon_{13}$ $\epsilon_{23}$.

The Bardet driver has the following format.

```plaintext
1  simulate constitutive testing BARDETMETHOD use material # <.>
2   scale_factor = <.>
3   series_file = <string>
4   sigma0 = ( <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> )
5   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.feioutput file is variable and material dependent.

The direct strain driver has the following format.

```plaintext
1 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 = <.>
```

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:

```plaintext
series_file = "increments.txt" where each line in increments.txt contains $d\epsilon_{11}$ $d\epsilon_{22}$ $d\epsilon_{33}$ $d\epsilon_{12}$ $d\epsilon_{13}$ $d\epsilon_{23}$.
```
### 1.6 List of Available Commands (tentative, not up to date)

1. `add acceleration field # <.> ax = <accel> ay = <accel> az = <accel>;`
2. `add constraint equal_dof with master node # <.> and slave node # <.> dof to constrain <.>;`
3. `add constraint equal_dof with node # <.> dof <.> master and node # <.> dof <.> slave;`
4. `add damping # <.> to element # <.>;`
5. `add damping # <.> to node # <.>;`
6. `add damping # <.> type Caughey3rd with a0 = <1/time> a1 = <time> a2 = <time>3 stiffness_to_use = <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;`
7. `add damping # <.> type Caughey4th with a0 = <1/time> a1 = <time> a2 = <time>5 a3 = <time>3 stiffness_to_use = <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;`
8. `add damping # <.> type Rayleigh with a0 = <1/time> a1 = <time> stiffness_to_use = <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;`
9. `add domain reduction method loading # <.> hdf5_file = <string> scale_factor = <.>;`
10. `add domain reduction method loading # <.> hdf5_file = <string>;`
11. `add element # <.> type 20NodeBrick using <.> Gauss points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.> material # <.>;`
12. `add element # <.> type 20NodeBrick with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.> use material # <.>;`
13. `add element # <.> type 20NodeBrick_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>;`
14. `add element # <.> type 20NodeBrick_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>;`
15. `add element # <.> type 20NodeBrick_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>;`
K_s = <stress> K_f = <stress>;
16 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 = ←<.> 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 # <.>;
20 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 = ←<.> 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 = ←<.> 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 = ←<.> 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>;
Jeremić et al. Real-ESSI Simulator 240 of 266

Real-ESSI Domain Specific Language version: March 30, 2020, 14:01

```plaintext
<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.> 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 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_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 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 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 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 >;
```

Real-ESSI Domain Specific Language version: March 30, 2020, 14:01
<stress> K_f = <stress>;  
38 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>;  
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>);  
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 = ← 
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 = ← 
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 (<.>, <.>) ← 
cross_section = <l^2> use material # <.>;  
47 add element # <.> type ShearBeam 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 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 = (<.>, <.>, ←
    <.>);
49 add element # <.> type truss with nodes (<.>, <.>) use material # <.> ←
    cross_section = <length^2> mass_density = <M/L^3> ;
50 add element # <.> type variable_node_brick_8_to_27 using <.> Gauss ←
    points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, ←
    <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ←
    <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>;
51 add elements (.<>) to physical_element_group "string";
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 magnitudes (<.>, <.>, ←
    <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>);
62 add load # <.> to element # <.> type surface at nodes (<.>, <.>, ←
    <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ←
    <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>);
63 add load # <.> to node # <.> type from_reactions;
add load # <.> to node # <.> type linear FORCETYPE = <force or moment>; //FORCETYPE = Fx Fy Fz Mx My Mz F_fluid_x F_fluid_y F_fluid_z
add load # <.> to node # <.> type path_series FORCETYPE = <force or moment> time_step = <time> series_file = "filename";
add load # <.> to node # <.> type path_time_series FORCETYPE = <force or moment> series_file = "filename";
add load # <.> to node # <.> type self_weight use acceleration field # <.>;
add mass to node # <.> mx = <mass> my = <mass> mz = <mass> Imx = <mass*length^2> Imy = <mass*length^2> Imz = <mass*length^2>;
add mass to node # <.> mx = <mass> my = <mass> mz = <mass>;
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 = <F/L^2> 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 = <.> 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 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_density>
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>

add material # <.> type linear_elastic_isotropic_3d
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 = <.>

add material # <.> type sanisand2004
mass_density = <M/L^3>
e0 = <.>
sanisand2004_G0 = <.>
poisson_ratio = <.>
sanisand2004_Pat = <stress>
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 = <.

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_lambda = <.
sanisand2008_xi = <.>
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_p0 = <stress>
sanisand2008_p_in = <.
algorithm = <explicit|implicit>
number_of_subincrements = <.>
maximum_number_of_iterations = <.>
tolerance_1 = <.>
tolerance_2 = <.>
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\>);
add material # <.> type uniaxial_elastic elastic_modulus = \(<F/L^2\> viscoelastic_modulus = \(<mass / length / time\>);
add material # <.> type uniaxial_steel01 yield_strength = \(<F/L^2\) elastic_modulus = \(<F/L^2\> strain_hardening_ratio = <.> a1 = <.> a2 = <.> a3 = <.> a4 = <.>;
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 = <.>;
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\>;
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\>;
add node # <.> at \((<length>,<length>,<length>)\) with <.> dofs;
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>.
102 define dynamic integrator Hilber_Hughes_Taylor with alpha = <.;>
103 define dynamic integrator Newmark with gamma = <.;> beta = <.;>
104 define load factor increment <.;>
105 define NDMaterial constitutive integration algorithm Forward_Euler;
106 define NDMaterial constitutive integration algorithm Forward_Euler_Subincrement number_of_subincrements =<.;>
107 define NDMaterial constitutive integration algorithm Forward_Euler | Forward_Euler_Subincrement | Backward_Euler | Backward_Euler_Subincrement yield_function_relative_tolerance = <.;> stress_relative_tolerance = <.;> maximum_iterations = <.;>
108 define physical_element_group "string";
109 define physical_node_group "string";
110 define solver ProfileSPD / UMFPack;
111 define static integrator displacement_control using node # <.;> dof DOFTYPE increment <length>;
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> ) 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> ) 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 = <.>
simulate constitutive testing undrained triaxial use material # <.>  
    strain_increment_size = <.> maximum_strain = <.>  
    number_of_times_reaching_maximum_strain = <.>
simulate using eigen algorithm number_of_modes = <.>
ux uy uz Ux Uy Uz rx ry rz;
while (.) { }
whos;
1.7 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)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a0</td>
</tr>
<tr>
<td>2</td>
<td>a1</td>
</tr>
<tr>
<td>3</td>
<td>a2</td>
</tr>
<tr>
<td>4</td>
<td>a3</td>
</tr>
<tr>
<td>5</td>
<td>a4</td>
</tr>
<tr>
<td>6</td>
<td>acceleration</td>
</tr>
<tr>
<td>7</td>
<td>acceleration_depth</td>
</tr>
<tr>
<td>8</td>
<td>acceleration_file</td>
</tr>
<tr>
<td>9</td>
<td>acceleration_filename</td>
</tr>
<tr>
<td>10</td>
<td>acceleration_scale_unit</td>
</tr>
<tr>
<td>11</td>
<td>add</td>
</tr>
<tr>
<td>12</td>
<td>algorithm</td>
</tr>
<tr>
<td>13</td>
<td>algorithm</td>
</tr>
<tr>
<td>14</td>
<td>all</td>
</tr>
<tr>
<td>15</td>
<td>all</td>
</tr>
<tr>
<td>16</td>
<td>allowed_subincrement_strain</td>
</tr>
<tr>
<td>17</td>
<td>alpha</td>
</tr>
<tr>
<td>18</td>
<td>alpha1</td>
</tr>
<tr>
<td>19</td>
<td>alpha2</td>
</tr>
<tr>
<td>20</td>
<td>and</td>
</tr>
<tr>
<td>21</td>
<td>angle</td>
</tr>
<tr>
<td>22</td>
<td>armstrong_frederick_cr</td>
</tr>
<tr>
<td>23</td>
<td>armstrong_frederick_ha</td>
</tr>
<tr>
<td>24</td>
<td>asynchronous</td>
</tr>
<tr>
<td>25</td>
<td>at</td>
</tr>
<tr>
<td>26</td>
<td>ax</td>
</tr>
<tr>
<td>27</td>
<td>axial_penalty_stiffness</td>
</tr>
<tr>
<td>28</td>
<td>axial_stiffness</td>
</tr>
<tr>
<td>29</td>
<td>axial_viscous_damping</td>
</tr>
<tr>
<td>30</td>
<td>ay</td>
</tr>
<tr>
<td>31</td>
<td>az</td>
</tr>
<tr>
<td>32</td>
<td>bending_Iy</td>
</tr>
<tr>
<td>33</td>
<td>bending_Iz</td>
</tr>
<tr>
<td>34</td>
<td>beta_min</td>
</tr>
<tr>
<td>35</td>
<td>case</td>
</tr>
<tr>
<td>36</td>
<td>cases</td>
</tr>
<tr>
<td>37</td>
<td>characteristic_strength</td>
</tr>
<tr>
<td>38</td>
<td>check</td>
</tr>
<tr>
<td>39</td>
<td>chi</td>
</tr>
<tr>
<td>40</td>
<td>cohesion</td>
</tr>
<tr>
<td>41</td>
<td>combine</td>
</tr>
<tr>
<td>42</td>
<td>compression</td>
</tr>
<tr>
<td>43</td>
<td>compressive_strength</td>
</tr>
<tr>
<td>44</td>
<td>compressive_yield_strength</td>
</tr>
<tr>
<td>45</td>
<td>compute</td>
</tr>
</tbody>
</table>
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 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  
lambda  
level  

line_search_beta
line_search_eta
line_search_max_iter
liquefaction_Alpha
liquefaction_c_h0
liquefaction.Dir
liquefaction_dre1
liquefaction_Dre2
liquefaction_EXPN
liquefaction_gamar
liquefaction_mdc
liquefaction_mfc
liquefaction_pa
liquefaction_pmin
load
load_factors_list
loading
local_y_vector
local_z_vector
M
M_in
magnitude
magnitudes
mass
mass_density
master
material
max_axial_stiffness
maximum_iterations
maximum_number_of_iterations
maximum_strain
maximum_stress
maximum_time_step
method
minimal
minimum_time_step
model
model
moment_x_stiffness
moment_y_stiffness
monotonic
motion
mu
mx
my
mz
name
NDMaterial
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
remove
residual_friction_coefficient
restart
restart_files
results
rho_a
rho_f
rho_s
rho_w
R0
RMC_e
RMC_eta0
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 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 to 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)

1 20NodeBrick
2 20NodeBrick_up
3 20NodeBrick_upU
4 27NodeBrick
5 27NodeBrick_up
6 27NodeBrick_upU
7 3NodeShell_ANDES
8 4NodeShell_ANDES
9 4NodeShell_MITC4
10 4NodeShell_NewMITC4
11 8_27_NodeBrick
12 8_27_NodeBrick_up
13 8_27_NodeBrick_upU
14 8NodeBrick
15 8NodeBrick_fluid_incompressible_up
16 8NodeBrick_up
17 8NodeBrick_upU
<table>
<thead>
<tr>
<th>Line</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Absolute_Norm_Displacement_Increment</td>
</tr>
<tr>
<td>19</td>
<td>Absolute_Norm_Unbalanced_Force</td>
</tr>
<tr>
<td>20</td>
<td>arclength_control</td>
</tr>
<tr>
<td>21</td>
<td>Average_Norm_Displacement_Increment</td>
</tr>
<tr>
<td>22</td>
<td>Average_Norm_Unbalanced_Force</td>
</tr>
<tr>
<td>23</td>
<td>Backward_Euler</td>
</tr>
<tr>
<td>24</td>
<td>BARDETMETHOD</td>
</tr>
<tr>
<td>25</td>
<td>beam_9dof_elastic</td>
</tr>
<tr>
<td>26</td>
<td>beam_displacement_based</td>
</tr>
<tr>
<td>27</td>
<td>beam_elastic</td>
</tr>
<tr>
<td>28</td>
<td>beam_elastic_lumped_mass</td>
</tr>
<tr>
<td>29</td>
<td>BeamColumnDispFiber3d</td>
</tr>
<tr>
<td>30</td>
<td>BearingElastomericPlasticity3d</td>
</tr>
<tr>
<td>31</td>
<td>BFGS</td>
</tr>
<tr>
<td>32</td>
<td>BondedContact</td>
</tr>
<tr>
<td>33</td>
<td>CamClay</td>
</tr>
<tr>
<td>34</td>
<td>Caughey3rd</td>
</tr>
<tr>
<td>35</td>
<td>Caughey4th</td>
</tr>
<tr>
<td>36</td>
<td>constant_mean_pressure_triaxial_strain_control</td>
</tr>
<tr>
<td>37</td>
<td>Cosserat8NodeBrick</td>
</tr>
<tr>
<td>38</td>
<td>Cosserat_linear_elastic_isotropic_3d</td>
</tr>
<tr>
<td>39</td>
<td>Cosserat_von_Mises</td>
</tr>
<tr>
<td>40</td>
<td>DIRECT_STRAIN</td>
</tr>
<tr>
<td>41</td>
<td>displacement_control</td>
</tr>
<tr>
<td>42</td>
<td>DOFTYPE</td>
</tr>
<tr>
<td>43</td>
<td>domain_reduction_method</td>
</tr>
<tr>
<td>44</td>
<td>drained_triaxial_strain_control</td>
</tr>
<tr>
<td>45</td>
<td>DruckerPrager</td>
</tr>
<tr>
<td>46</td>
<td>DruckerPragerArmstrongFrederickLE</td>
</tr>
<tr>
<td>47</td>
<td>DruckerPragerArmstrongFrederickNE</td>
</tr>
<tr>
<td>48</td>
<td>DruckerPragerMultipleYieldSurface</td>
</tr>
<tr>
<td>49</td>
<td>DruckerPragerMultipleYieldSurfaceGoverGmax</td>
</tr>
<tr>
<td>50</td>
<td>DruckerPragerNonAssociateArmstrongFrederick</td>
</tr>
<tr>
<td>51</td>
<td>DruckerPragerNonAssociateLinearHardening</td>
</tr>
<tr>
<td>52</td>
<td>DruckerPragervonMises</td>
</tr>
<tr>
<td>53</td>
<td>dynamic</td>
</tr>
<tr>
<td>54</td>
<td>eigen</td>
</tr>
<tr>
<td>55</td>
<td>elastic3d</td>
</tr>
<tr>
<td>56</td>
<td>Elastic_Membrane_Plate</td>
</tr>
<tr>
<td>57</td>
<td>ElasticFourNodeQuad</td>
</tr>
<tr>
<td>58</td>
<td>Energy_Increment</td>
</tr>
<tr>
<td>59</td>
<td>equal_dof</td>
</tr>
<tr>
<td>60</td>
<td>F_fluid_x</td>
</tr>
<tr>
<td>61</td>
<td>F_fluid_y</td>
</tr>
<tr>
<td>62</td>
<td>F_fluid_z</td>
</tr>
<tr>
<td>63</td>
<td>FiberSection</td>
</tr>
<tr>
<td>64</td>
<td>ForceBasedCoupledHardContact</td>
</tr>
<tr>
<td>65</td>
<td>ForceBasedCoupledSoftContact</td>
</tr>
<tr>
<td>66</td>
<td>ForceBasedElasticContact</td>
</tr>
<tr>
<td>67</td>
<td>ForceBasedHardContact</td>
</tr>
<tr>
<td>68</td>
<td>ForceBasedSoftContact</td>
</tr>
<tr>
<td>69</td>
<td>FORCETYPE</td>
</tr>
</tbody>
</table>
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
112 Relative_Energy_Increment
113 Relative_Norm_Displacement_Increment
114 Relative_Norm_Unbalance
115 Relative_Norm_Unbalanced_Force
116 roundedMohrCoulomb
117 RoundedMohrCoulombMultipleYieldSurface
118 sanisand2004
119 sanisand2004_legacy
120 sanisand2008
121 self_weight

Real-ESI Domain Specific Language  version: March 30, 2020, 14:01
ShearBeam
solid fluid interaction transient
static
StressBasedCoupledHardContact_ElPPlShear
StressBasedCoupledHardContact_NonLinHardShear
StressBasedCoupledHardContact_NonLinHardSoftShear
StressBasedCoupledSoftContact
StressBasedCoupledSoftContact_ElPPlShear
StressBasedCoupledSoftContact_NonLinHardShear
StressBasedCoupledSoftContact_NonLinHardSoftShear
SuperElementLinearElasticImport
support
surface
transient
truss
TsinghuaLiquefactionModelCirclePiPlane
TsinghuaLiquefactionModelNonCirclePiPlane
UMFPack
undrained simple shear
undrained triaxial
undrained triaxial stress control
uniaxial_concrete02
uniaxial_elastic
uniaxial_steel01
uniaxial_steel02
variable transient
variable_node_brick_8_to_27
vonMises
vonMisesArmstrongFrederick
vonMisesMultipleYieldSurface
vonMisesMultipleYieldSurfaceGoverGmax
With_no_convergence_check
beta
gamma
delta
ux
uy
uz
rx
ry
rz
Ux
Uy
Uz
174  p
175  M
176  m
177  kg
178  s
179  cm
180  mm
181  km
182  Hz
183  Minute
184  Hour
185  Day
186  Week
187  ms
188  ns
189  N
190  kN
191  Pa
192  kPa
193  MPa
194  GPa
195  pound
196  lbm
197  lbf
198  inch
199  in
200  feet
201  ft
202  yard
203  mile
204  psi
205  ksi
206  kip
207  g
208  pi
209
210  NUMBER_OF_NODES
211  NUMBER_OF_ELEMENTS
212  CURRENT_TIME
213  NUMBER_OF_SP_CONSTRAINTS
214  NUMBER_OF_MP_CONSTRAINTS
215  NUMBER_OF_LOADS
216  IS_PARALLEL
217  SIMULATE_EXIT_FLAG
218  then
219  while
220  do
221  let
222  vector
223
224  cos
225  sin
<table>
<thead>
<tr>
<th></th>
<th>Function</th>
</tr>
</thead>
<tbody>
<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>
1.8 Integrated Development Environment (IDE) for DSL
1.9 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).
1.10 Model Development and Mesh Generation using gmesh
1.11 Model Input File Editing using Sublime

http://www.sublimetext.com/


