source time function is used (Aki and Richards, 2002). Nine recording points are set as recording stations (Figure 706.1). Direction of the fault is aligned parallel to the north (strike = 0°) and recording station azimuth is set to 0°, 45°, and 90°.

Figure 706.2 – 706.10 show analyses results for the example. Legends on figures mean 'component (epicentral distance, receiver depth)'. EW, NS, and UD components mean East - West, North - South, and Up - Down, respectively (those terms are used for all seismograms, hereafter).

Only EW components are predicted on the stations at 0° azimuth (Figure 706.2 – 706.4) and NS components are showed up on the station at 90° azimuth (Figure 706.8 – 706.10). On the station at 45° azimuth, all components (EW, NS, and UD) are observed.
Figure 706.2: Calculated time history acceleration, station azimuth = 0°
Figure 706.3: Calculated time history velocity, station azimuth = 0°.
Figure 706.4: Calculated time history displacement, station azimuth = 0°.
Figure 706.5: Calculated time history acceleration, station azimuth = 45°
Figure 706.6: Calculated time history velocity, station azimuth = 45°.
Figure 706.7: Calculated time history displacement, station azimuth = 45°
Figure 706.8: Calculated time history acceleration, station azimuth = 90°
Figure 706.9: Calculated time history velocity, station azimuth = 90°
Figure 706.10: Calculated time history displacement, station azimuth = 90°
Case 2: dip-slip fault / single layer ground

Similar fault is tested for vertical (dip) slip (rake = 90°) fault case. Figure 706.11 shows model used for the analysis. Ground, fault, and wave properties are shown as below.

**Figure 706.11:** Ground and fault model used for analysis, results are captured on circles

- **Ground properties**
  - $V_S = 1 \text{ km/s}$
  - $V_P/V_S = 1.73$
  - Poisson’s ratio = 0.25
  - Density = 1.32 g/cm$^3$
  - Shear modulus = 1.32 GPa
  - Elastic modulus = 3.31 GPa

- **Fault properties**
  - Moment magnitude = 3.5
  - Strike = 0°
  - Dip = 90°
  - Rake = 90°
  - Double - coupled source
  - Triangular source time function

- **Wave properties**
\[ \text{dt} = 0.1 \text{ s} \text{ (Max available freq. = 5 Hz, Nyquist freq.)} \]

Similar to the strike slip example, single layer ground \((V_s = 1 \text{ km/s})\) is modeled. Fault is located at 2 km depth, 2 km away from the recording stations. Double coupled fault source is assumed and triangular source time function is used (Aki and Richards, 2002). Nine recording points are set as recording stations (Figure 706.11). Direction of the fault is aligned parallel to the north (strike = 0°) and rake is 90°. Station azimuth is set to 0°, 45°, and 90°.

Figure 706.12 – 706.20 show analyses results for this example. Since it’s dip slip case, permanent deformation on UD components are observed (Figure 706.17 and 706.20).
Figure 706.12: Calculated time history acceleration, station azimuth = 0°.
Figure 706.13: Calculated time history velocity, station azimuth = 0°.
Figure 706.14: Calculated time history displacement, station azimuth = 0°.
Figure 706.15: Calculated time history acceleration, station azimuth = 45°
Figure 706.16: Calculated time history velocity, station azimuth = 45°
Figure 706.17: Calculated time history displacement, station azimuth = 45°
Figure 706.18: Calculated time history acceleration, station azimuth = 90°.
Figure 706.19: Calculated time history velocity, station azimuth = 90°
Figure 706.20: Calculated time history displacement, station azimuth = 90°
Case 3: normal fault / single layer ground

Figure 706.21: Ground and fault model used for analysis, results are captured on circles

Normal fault is tested. Figure 706.21 shows model used for the analysis. Wave is propagated through the single layer ground ($V_s = 1 \text{ km/s}$). Properties are shown as below.

- **Ground properties**
  - $V_s = 1 \text{ km/s}$
  - $V_P/V_S = 1.73$
  - Poisson’s ratio = 0.25
  - Density = 1.32 g/cm$^3$
  - Shear modulus = 1.32 GPa
  - Elastic modulus = 3.31 GPa

- **Fault properties**
  - Moment magnitude = 5.0
  - Strike = 0$^\circ$
  - Dip = 45$^\circ$
  - Rake = 90$^\circ$
  - Double-coupled source
  - Triangular source time function

- **Wave properties**
\[ dt = 0.1 \text{ s (Max available freq. = 5 Hz, Nyquist freq.)} \]

In this example, the distance between the fault and the station is increased and magnitude is changed also (Mw = 5.0). Fault is located at 30 km depth, 30 km away from the recording stations (Figure 706.21). Double coupled fault source is assumed and triangular source time function is used (Aki and Richards, 2002). Recording points are similar as prior examples (total 9 stations). Azimuth of recording station is set to 0°, 45°, and 90°.

Figure 706.22 – 706.30 show analyses results. Since the distance between fault and station is increased to 30 km and waves are propagated through the ground with relatively low shear wave velocity, arrival of propagating and reflecting waves can be observed easily (the first arrival of P wave followed by S wave). Permanent displacements by the fault movement are observed as desired at all stations (0°, 45°, and 90°).
Figure 706.22: Calculated time history acceleration, station azimuth = 0°.
Figure 706.23: Calculated time history velocity, station azimuth = 0°.
Figure 706.24: Calculated time history displacement, station azimuth = 0°.
Figure 706.25: Calculated time history acceleration, station azimuth = 45°
Figure 706.26: Calculated time history velocity, station azimuth = 45°
Figure 706.27: Calculated time history displacement, station azimuth = 45°
Figure 706.28: Calculated time history acceleration, station azimuth = 90°
Figure 706.29: Calculated time history velocity, station azimuth = 90°
Figure 706.30: Calculated time history displacement, station azimuth = 90°
Case 4: normal fault / layered ground

Normal fault within layered ground is modeled. Model is as shown Figure 706.31 and 706.32. Properties are shown below.

- Ground properties: see Figure 706.32 and Table 706.1
- Fault properties
- Moment magnitude = 5.0
- Strike = 0°
- Dip = 45°
- Rake = 90°
- Double - coupled source
- Triangular source time function

• Wave properties
  - \( dt = 0.1 \text{ s} \) (Max available freq. = 5 Hz, Nyquist freq.)

Fault is located at depth of 30 km, 30 km away from the recording point (station) at surface. Double coupled fault source is assumed and triangular source time function is used (Aki and Richards, 2002). Recording points are similar as prior examples (total 9 stations). The 1D standard southern California model (Hadley and Kanamori (1977), \( h_k \) model hereafter) is used for ground layering. As shown in Figure 706.32, \( h_k \) model is interpolated and divided to define ground layer (Table 706.1).

Results are shown on Figure 706.33 – 706.35. Since wave is propagated through the layered ground, compared to prior examples, more realistic waves are observed.
### Table 706.1: Ground properties for the example

<table>
<thead>
<tr>
<th>Depth (km)</th>
<th>Thickness (km)</th>
<th>VS (km/s)</th>
<th>VP/VS</th>
<th>QB (km/s)</th>
<th>VP (GPa)</th>
<th>Poisson’s R</th>
<th>Density (g/cm³)</th>
<th>G (GPa)</th>
<th>E (GPa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.50</td>
<td>1.730</td>
<td>500</td>
<td>0.87</td>
<td>0.25</td>
<td>1.05</td>
<td>0.27</td>
<td>0.67</td>
</tr>
<tr>
<td>0.02</td>
<td>0.01</td>
<td>0.80</td>
<td>1.730</td>
<td>500</td>
<td>1.38</td>
<td>0.25</td>
<td>1.21</td>
<td>0.77</td>
<td>1.93</td>
</tr>
<tr>
<td>0.03</td>
<td>0.01</td>
<td>0.97</td>
<td>1.730</td>
<td>500</td>
<td>1.68</td>
<td>0.25</td>
<td>1.31</td>
<td>1.23</td>
<td>3.07</td>
</tr>
<tr>
<td>0.04</td>
<td>0.01</td>
<td>1.09</td>
<td>1.730</td>
<td>500</td>
<td>1.89</td>
<td>0.25</td>
<td>1.37</td>
<td>1.64</td>
<td>4.09</td>
</tr>
<tr>
<td>0.05</td>
<td>0.01</td>
<td>1.19</td>
<td>1.730</td>
<td>500</td>
<td>2.05</td>
<td>0.25</td>
<td>1.43</td>
<td>2.01</td>
<td>5.01</td>
</tr>
<tr>
<td>0.06</td>
<td>0.01</td>
<td>1.26</td>
<td>1.730</td>
<td>500</td>
<td>2.19</td>
<td>0.25</td>
<td>1.47</td>
<td>2.34</td>
<td>5.86</td>
</tr>
<tr>
<td>0.07</td>
<td>0.01</td>
<td>1.33</td>
<td>1.730</td>
<td>500</td>
<td>2.30</td>
<td>0.25</td>
<td>1.51</td>
<td>2.66</td>
<td>6.64</td>
</tr>
<tr>
<td>0.08</td>
<td>0.01</td>
<td>1.38</td>
<td>1.730</td>
<td>500</td>
<td>2.40</td>
<td>0.25</td>
<td>1.54</td>
<td>2.95</td>
<td>7.36</td>
</tr>
<tr>
<td>0.09</td>
<td>0.01</td>
<td>1.43</td>
<td>1.730</td>
<td>500</td>
<td>2.48</td>
<td>0.25</td>
<td>1.56</td>
<td>3.22</td>
<td>8.04</td>
</tr>
<tr>
<td>0.1</td>
<td>0.01</td>
<td>1.48</td>
<td>1.730</td>
<td>500</td>
<td>2.56</td>
<td>0.25</td>
<td>1.59</td>
<td>3.48</td>
<td>8.69</td>
</tr>
<tr>
<td>0.2</td>
<td>0.10</td>
<td>1.77</td>
<td>1.730</td>
<td>500</td>
<td>3.07</td>
<td>0.25</td>
<td>1.75</td>
<td>5.50</td>
<td>13.75</td>
</tr>
<tr>
<td>0.3</td>
<td>0.10</td>
<td>1.94</td>
<td>1.730</td>
<td>500</td>
<td>3.36</td>
<td>0.25</td>
<td>1.85</td>
<td>6.98</td>
<td>17.44</td>
</tr>
<tr>
<td>0.4</td>
<td>0.10</td>
<td>2.07</td>
<td>1.730</td>
<td>500</td>
<td>3.57</td>
<td>0.25</td>
<td>1.91</td>
<td>8.17</td>
<td>20.41</td>
</tr>
<tr>
<td>0.5</td>
<td>0.10</td>
<td>2.16</td>
<td>1.730</td>
<td>500</td>
<td>3.74</td>
<td>0.25</td>
<td>1.97</td>
<td>9.18</td>
<td>22.93</td>
</tr>
<tr>
<td>0.6</td>
<td>0.10</td>
<td>2.24</td>
<td>1.730</td>
<td>500</td>
<td>3.87</td>
<td>0.25</td>
<td>2.01</td>
<td>10.06</td>
<td>25.13</td>
</tr>
<tr>
<td>0.7</td>
<td>0.10</td>
<td>2.30</td>
<td>1.730</td>
<td>500</td>
<td>3.98</td>
<td>0.25</td>
<td>2.04</td>
<td>10.85</td>
<td>27.10</td>
</tr>
<tr>
<td>0.8</td>
<td>0.10</td>
<td>2.36</td>
<td>1.730</td>
<td>500</td>
<td>4.08</td>
<td>0.25</td>
<td>2.08</td>
<td>11.56</td>
<td>28.88</td>
</tr>
<tr>
<td>0.9</td>
<td>0.10</td>
<td>2.41</td>
<td>1.730</td>
<td>500</td>
<td>4.17</td>
<td>0.25</td>
<td>2.10</td>
<td>12.21</td>
<td>30.51</td>
</tr>
<tr>
<td>1.0</td>
<td>0.10</td>
<td>2.45</td>
<td>1.730</td>
<td>500</td>
<td>4.25</td>
<td>0.25</td>
<td>2.13</td>
<td>12.82</td>
<td>32.02</td>
</tr>
<tr>
<td>1.5</td>
<td>0.50</td>
<td>2.63</td>
<td>1.730</td>
<td>500</td>
<td>4.54</td>
<td>0.25</td>
<td>2.22</td>
<td>15.33</td>
<td>38.30</td>
</tr>
<tr>
<td>2.5</td>
<td>0.00</td>
<td>2.84</td>
<td>1.730</td>
<td>500</td>
<td>4.92</td>
<td>0.25</td>
<td>2.34</td>
<td>18.92</td>
<td>47.28</td>
</tr>
<tr>
<td>3.5</td>
<td>0.00</td>
<td>2.98</td>
<td>1.730</td>
<td>500</td>
<td>5.16</td>
<td>0.25</td>
<td>2.42</td>
<td>21.57</td>
<td>53.89</td>
</tr>
<tr>
<td>4.5</td>
<td>0.00</td>
<td>3.09</td>
<td>1.730</td>
<td>500</td>
<td>5.35</td>
<td>0.25</td>
<td>2.48</td>
<td>23.70</td>
<td>59.21</td>
</tr>
<tr>
<td>5.5</td>
<td>0.00</td>
<td>3.18</td>
<td>1.730</td>
<td>600</td>
<td>5.50</td>
<td>0.25</td>
<td>2.53</td>
<td>25.59</td>
<td>63.93</td>
</tr>
<tr>
<td>16.0</td>
<td>0.50</td>
<td>3.64</td>
<td>1.731</td>
<td>600</td>
<td>6.30</td>
<td>0.25</td>
<td>2.79</td>
<td>36.92</td>
<td>92.26</td>
</tr>
</tbody>
</table>
Figure 706.33: Calculated time history acceleration, station azimuth $= 0^\circ$. 
Figure 706.34: Calculated time history velocity, station azimuth = 0°.
Figure 706.35: Calculated time history displacement, station azimuth = 0°.
Figure 706.36: Calculated time history acceleration, station azimuth = 45°
Figure 706.37: Calculated time history velocity, station azimuth = 45°
Figure 706.38: Calculated time history displacement, station azimuth = 45°
Figure 706.39: Calculated time history acceleration, station azimuth = 90°
Figure 706.40: Calculated time history velocity, station azimuth = 90°
Figure 706.41: Calculated time history displacement, station azimuth = 90°
Case 5: Northridge earthquake / layered ground

In this example, Northridge earthquakes are simulated. Northridge earthquake has occurred on January, 1994 with a moment magnitude of 6.7. Properties are shown as below and partly adapted from Hisada (2008) and Wald et al. (1996).

- Ground properties: see Table 706.2

- Fault properties
  - Moment magnitude = 6.7
  - Strike = 122°
  - Dip = 40°
  - Rake = 140°
  - Double - coupled source
  - Triangular source time function

- Wave properties
  - $dt = 0.05 \text{ s (Max available freq. } = 10 \text{ Hz, Nyquist freq.)}$

Source depth is set as 25 km and epicentral distance from the fault to the station is set as 30 km. Station is located on the ground surface. As shown in Table 706.2, ground is divided into 9 layers (Hisada, 2008).

Figure 706.42 shows analyses results. Computed results are compared with measured one. Measured records are obtained from cosmos virtual data center http://db.cosmos-eq.org/. As shown in Figure 706.42, predicted seismogram agrees well with measured ones considering the fk package assumes a single point source and simplified ground.pdf.
Table 706.2: Ground properties for the example

<table>
<thead>
<tr>
<th>Depth (km)</th>
<th>Thickness (km)</th>
<th>VS (km/s)</th>
<th>VP/VS</th>
<th>QB (km/s)</th>
<th>VP (km/s)</th>
<th>Poisson's R</th>
<th>Density (g/cm³)</th>
<th>G (GPa)</th>
<th>E (GPa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.05</td>
<td>0.30</td>
<td>1.730</td>
<td>600</td>
<td>0.52</td>
<td>0.25</td>
<td>0.94</td>
<td>0.08</td>
<td>0.21</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>0.40</td>
<td>1.730</td>
<td>600</td>
<td>0.69</td>
<td>0.25</td>
<td>0.99</td>
<td>0.16</td>
<td>0.40</td>
</tr>
<tr>
<td>0.2</td>
<td>0.10</td>
<td>0.50</td>
<td>1.730</td>
<td>600</td>
<td>0.87</td>
<td>0.25</td>
<td>1.05</td>
<td>0.26</td>
<td>0.65</td>
</tr>
<tr>
<td>0.3</td>
<td>0.10</td>
<td>0.75</td>
<td>1.730</td>
<td>600</td>
<td>1.30</td>
<td>0.25</td>
<td>1.19</td>
<td>0.67</td>
<td>1.67</td>
</tr>
<tr>
<td>0.5</td>
<td>0.20</td>
<td>1.00</td>
<td>1.730</td>
<td>600</td>
<td>1.73</td>
<td>0.25</td>
<td>1.32</td>
<td>1.32</td>
<td>3.31</td>
</tr>
<tr>
<td>1.5</td>
<td>1.00</td>
<td>2.00</td>
<td>1.730</td>
<td>600</td>
<td>3.46</td>
<td>0.25</td>
<td>1.88</td>
<td>7.51</td>
<td>18.76</td>
</tr>
<tr>
<td>4</td>
<td>2.50</td>
<td>3.20</td>
<td>1.730</td>
<td>600</td>
<td>5.54</td>
<td>0.25</td>
<td>2.54</td>
<td>26.03</td>
<td>65.02</td>
</tr>
<tr>
<td>27</td>
<td>23.00</td>
<td>3.60</td>
<td>1.730</td>
<td>600</td>
<td>6.23</td>
<td>0.25</td>
<td>2.76</td>
<td>35.81</td>
<td>89.46</td>
</tr>
<tr>
<td>40</td>
<td>13.00</td>
<td>3.90</td>
<td>1.730</td>
<td>900</td>
<td>6.75</td>
<td>0.25</td>
<td>2.93</td>
<td>44.55</td>
<td>111.30</td>
</tr>
</tbody>
</table>
Figure 706.42: Analysis result, thick line is observed (SYL station, http://db.cosmos-eq.org/); results presented by the thin line are calculated using $f_k/syn$. 

Jeremić et al., UCD and LBNL version: 3. August, 2020, 19:54
Appendix 707


(In collaboration with Dr. Yuan Feng, Prof. José Abell, Mr. Sumeet Kumar Sinha, Dr. Han Yang and Mr. Hexiang Wang)
This chapter presents a number of illustrative examples. The main aim is simple: present Real-ESSI Simulator features (available elements, algorithms, domain specific language (DSL), &c.) through a number of simple examples. It is noted that all presented elements and algorithms work in sequential and parallel mode. However, presented examples are very small, and parallel mode will not bring any benefits.
707.1 Elastic Beam Element Under Static Loading

This is a simple beam example under static loading in three directions. The diagram below shows the loading in one bending direction.

![Diagram of a cantilever beam with a downward force at one end.](attachment:image)

**ESSI model fei/DSL file:**

```plaintext
1 model name "beam_1element" ;
2 // define the node coordinates
3 add node # 1 at ( 0.0*m , 0.0*m, 0.0*m) with 6 dofs;
4 add node # 2 at ( 1.0*m , 0.0*m, 0.0*m) with 6 dofs;
5 // Geometry: width and height. Help the beam definition.
6 b=0.2*m;
7 h=0.2*m;
8 I=b*h^3/12.0;
9 // define the beam element
10 add element # 1 type beam_elastic with nodes (1,2)
11 cross_section = b*h
12 elastic_modulus = 1e9*N/m^2
13 shear_modulus = 5e8*N/m^2
14 torsion_Jx = 0.33*b*h^3
15 bending_Iy = I
16 bending_Iz = I
17 mass_density = 0*kg/m^3
18 xz_plane_vector = ( 1, 0, 1)
19 joint_1_offset = (0*m, 0*m, 0*m)
20 joint_2_offset = (0*m, 0*m, 0*m);
21 // add boundary condition
22 fix node # 1 dofs all;
23 // axial loading
24 new loading stage "axial";
25 add load # 1 to node # 2 type linear Fx = 1*N;
26 define load factor increment 1;
27 define algorithm With_no_convergence_check ;
28 define solver ProfileSPD;
29 simulate 1 steps using static algorithm;
```
// bending in one direction
new loading stage "bending1";
remove load # 1;
add load # 2 to node # 2 type linear Fy = 1*N;
define load factor increment 1;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 1 steps using static algorithm;

// bending in the other direction
new loading stage "bending2";
remove load # 2;
add load # 3 to node # 2 type linear Fz = 1*N;
define load factor increment 1;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 1 steps using static algorithm;

bye;

The ESSI model fei/DSL files for this example can be downloaded [here](#).
707.2 Elastic Beam Element under Dynamic Loading

Problem description:

![Figure 707.2: The cantilever model.](image)

**ESSI model fei/DSL file:**

```plaintext
model name "beam_1element" ;

// add node
add node # 1 at ( 0.0*m , 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at ( 1.0*m , 0.0*m, 0.0*m) with 6 dofs;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
possion_ratio=0.3;

// add elements
add element # 1 type beam_elastic with nodes (1,2)
cross_section = b*h
elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2/(1+possion_ratio)
torsion_Jx = 0.33*b*h^3
bending_Iy = b*h^3/12
bending_Iz = b*h^3/12
mass_density = rho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
```

 Jeremić et al., UCD and LBNL version: 3. August, 2020, 19:54
joint_2_offset = (0*m, 0*m, 0*m);

// add boundary condition
fix node # 1 dofs all;

// // --slowLoading---------------------------------------------------------------
// add load in 180 seconds. (Slow)
// new loading stage "slowLoading";
// add load # 1 to node # 2 type path_time_series
// Fz = 1.*N
// series_file = "slowLoading.txt" ;
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check ;
// define solver ProfileSPD;
// simulate 2000 steps using transient algorithm
// time_step = 0.1*s;

// // --fastLoading---------------------------------------------------------------
// add load in 0.6 seconds (Fast)
// remove load # 1;
// new loading stage "fastLoading";
// add load # 2 to node # 2 type path_time_series
// Fz = 1.*N
// series_file = "fastLoading.txt" ;
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check ;
// define solver ProfileSPD;
// simulate 1000 steps using transient algorithm
// time_step = 0.01*s;

// // --freeVibration-------------------------------------------------------------
// add a load and then release to free vibration
// remove load # 2;
new loading stage "freeVibration";
add load # 3 to node # 2 type path_time_series
 Fz = 1.*N
 series_file = "freeVibration.txt" ;
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 2000 steps using transient algorithm
 time_step = 0.01*s;

bye;
Displacement Results

Figure 707.3: Slow loading condition, vertical displacements of the cantilever tip.

Figure 707.4: Fast loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.5: Free vibration, vertical displacements of the cantilever tip.
707.3 Cantilever, 5 Elastic Beam Elements

Problem description:

![Diagram of the cantilever model](image)

Figure 707.6: The cantilever model.

ESSI model fei/DSL file:

```plaintext
model name "beam_5element";

// add node
add node # 1 at (0.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at (0.2*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 3 at (0.4*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 4 at (0.6*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 5 at (0.8*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 6 at (1.0*m, 0.0*m, 0.0*m) with 6 dofs;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
p vosson_ratio=0.3;

// Cross section geometry: width and height
b=0.2*m;
h=0.2*m;

// add elements
ii=1;
while (ii<6) {
```

add element # ii type beam_elastic with nodes (ii,ii+1)
cross_section = b*h
elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2/(1+poission_ratio)
torsion_Jx = 0.33*b*h^3
bending_Iy = b*h^3/12
bending_Iz = b*h^3/12
mass_density = rho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
ii+=1;
}

// add boundary condition
fix node # 1 dofs all;

// // --slowLoading---------------------------------------------------------------
// // add load in 180 seconds.
// // new loading stage "slowLoading";
// add load # 1 to node # 6 type path_time_series
// Fz = 1.*N
// series_file = "slowLoading.txt" ;
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check ;
// define solver ProfileSPD;
// simulate 2000 steps using transient algorithm
// time_step = 0.1*s;

// // --fastLoading---------------------------------------------------------------
// add load in 0.6 seconds.
// remove load # 1;
// new loading stage "fastLoading";
// add load # 2 to node # 6 type path_time_series
// Fz = 1.*N
// series_file = "fastLoading.txt" ;
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check ;
// define solver ProfileSPD;
// simulate 1000 steps using transient algorithm
// time_step = 0.01*s;

// // --freeVibration-------------------------------------------------------------
// add a load and then release for free vibration
// remove load # 2;
new loading stage "freeVibration"
add load # 3 to node # 6 type path_time_series
Fz = 1.*N
series_file = "freeVibration.txt" ;
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.1*s;
bye;

Displacement results

![Figure 707.7: Slow loading condition, vertical displacements of the cantilever tip.](image)

The ESSI model fei/DSL files for this example can be downloaded [here](link).
Figure 707.8: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.9: Free vibration condition, vertical displacements of the cantilever tip.
707.4 Cantilever, One 27 Node Brick Element, Dynamic Loading

Problem description:

![Cantilever Model](image)

Figure 707.10: The cantilever model.

**ESSI model fei/DSL file:**

```plaintext
model name "brick_1element" ;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
povsion_ratio=0.3;

add material # 1 type linear_elastic_isotropic_3d_LT
mass_density = rho
elastic_modulus = elastic_constant
poisson_ratio = povsion_ratio;

add node # 1 at ( 0.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 2 at ( 0.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 3 at ( 1.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 4 at ( 1.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 5 at ( 0.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 6 at ( 1.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 7 at ( 0.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 8 at ( 0.0000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 9 at ( 0.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
```

---

**Jeremić et al., UCD and LBNL**

**version:** 3. August, 2020, 19:54
add node # 10 at (0.5000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 11 at (1.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
add node # 12 at (0.5000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 13 at (0.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add node # 14 at (0.5000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 15 at (1.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
add node # 16 at (0.5000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 17 at (0.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 18 at (0.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 19 at (1.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 20 at (1.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 21 at (0.5000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 22 at (0.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 23 at (0.5000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
add node # 24 at (1.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
add node # 25 at (0.5000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 26 at (0.5000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
add node # 27 at (0.5000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;

add element # 1 type 27NodeBrickLT with nodes(2, 1, 3, 4, 5, 8, 7, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27) use material # 1;

fix node # 1 dofs all;
fix node # 2 dofs all;
fix node # 5 dofs all;
fix node # 8 dofs all;
fix node # 9 dofs all;
fix node # 13 dofs all;
fix node # 17 dofs all;
fix node # 18 dofs all;
fix node # 22 dofs all;

// // ---slowLoading-----------------------------------------------
// new loading stage "slowLoading"
// add load # 1 to node # 4 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
// add load # 2 to node # 6 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
// add load # 3 to node # 3 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
// add load # 4 to node # 7 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
// add load # 5 to node # 20 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
// add load # 6 to node # 11 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
// add load # 7 to node # 15 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
"slowLoading.txt";
75  // add load # 8 to node # 19 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
76  // add load # 9 to node # 24 type path_time_series Fz=4/9.0*N series_file = "slowLoading.txt";
77  // add algorithm and solver
78  // define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
79  // define algorithm With_no_convergence_check;
80  // define solver ProfileSPD;
81  // simulate 2000 steps using transient algorithm
82  // time_step = 0.1*s;
83
84  // // --fastLoading---------------------------------------------------------------
85  // new loading stage "fastLoading";
86  // add load # 101 to node # 4 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
87  // add load # 102 to node # 6 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
88  // add load # 103 to node # 3 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
89  // add load # 104 to node # 7 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
90  // add load # 105 to node # 20 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
91  // add load # 106 to node # 11 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
92  // add load # 107 to node # 15 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
93  // add load # 108 to node # 19 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
94  // add load # 109 to node # 24 type path_time_series Fz=4/9.0*N series_file = "fastLoading.txt";
95  // add algorithm and solver
96  // define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
97  // define algorithm With_no_convergence_check;
98  // define solver ProfileSPD;
99  // simulate 1000 steps using transient algorithm
100  // time_step = 0.01*s;
101
102  // // --freeVibration---------------------------------------------------------------
103  // new loading stage "freeVibration";
104  // add load # 201 to node # 4 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
105  // add load # 202 to node # 6 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
106  // add load # 203 to node # 3 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
add load # 204 to node # 7 type path_time_series Fz=1/36.0*N series_file = \"freeVibration.txt\";
add load # 205 to node # 20 type path_time_series Fz=1/9.0*N series_file = \"freeVibration.txt\";
add load # 206 to node # 11 type path_time_series Fz=1/9.0*N series_file = \"freeVibration.txt\";
add load # 207 to node # 15 type path_time_series Fz=1/9.0*N series_file = \"freeVibration.txt\";
add load # 208 to node # 19 type path_time_series Fz=1/9.0*N series_file = \"freeVibration.txt\";
add load # 209 to node # 24 type path_time_series Fz=4/9.0*N series_file = \"freeVibration.txt\";
// add algorithm and solver
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 10000 steps using transient algorithm
time_step = 0.001*s;
// end
bye;

Displacement results against time series

![Graph](image)

Figure 707.11: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded [here](#).
Figure 707.12: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.13: Free vibration condition, vertical displacements of the cantilever tip.
707.5 Simulate Cantilever Using Five 27 Node Brick Elements

Problem description:

![Cantilever Diagram]

Figure 707.14: The cantilever model.

ESSI model fei/DSL file:

```plaintext
model name "brick_5element" ;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
 rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);

add material # 1 type linear_elastic_isotropic_3d_LT
mass_density = rho
elastic_modulus = elastic_constant
poisson_ratio = possiion_ratio;

add node # 1 at (0.0*m, 0.0*m , 0.0*m) with 3 dofs;
add node # 2 at (0.1*m, 0.0*m , 0.0*m) with 3 dofs;
add node # 3 at (0.2*m, 0.0*m , 0.0*m) with 3 dofs;
add node # 4 at (0.0*m, 0.1*m , 0.0*m) with 3 dofs;
add node # 5 at (0.1*m, 0.1*m , 0.0*m) with 3 dofs;
add node # 6 at (0.2*m, 0.1*m , 0.0*m) with 3 dofs;
add node # 7 at (0.0*m, 0.2*m , 0.0*m) with 3 dofs;
add node # 8 at (0.1*m, 0.2*m , 0.0*m) with 3 dofs;
add node # 9 at (0.2*m, 0.2*m , 0.0*m) with 3 dofs;
```

Jeremić et al., Real-ESSI
32
33  fix node No 1 dofs ux uy uz;
34  fix node No 2 dofs ux uy uz;
35  fix node No 3 dofs ux uy uz;
36  fix node No 4 dofs ux uy uz;
37  fix node No 5 dofs ux uy uz;
38  fix node No 6 dofs ux uy uz;
39  fix node No 7 dofs ux uy uz;
40  fix node No 8 dofs ux uy uz;
41  fix node No 9 dofs ux uy uz;
42  e = 0;
43  hh = 0*m;
44  NBricks=5;
45  dz = 0.2*m;
46  while ( e < NBricks)
47  {
48    hh += dz;
49    add node # 10+18*e at (0.0*m, 0.0*m, hh - 0.5*dz) with 3 dofs;
50    add node # 11+18*e at (0.1*m, 0.0*m, hh - 0.5*dz) with 3 dofs;
51    add node # 12+18*e at (0.2*m, 0.0*m, hh - 0.5*dz) with 3 dofs;
52    add node # 13+18*e at (0.0*m, 0.1*m, hh - 0.5*dz) with 3 dofs;
53    add node # 14+18*e at (0.1*m, 0.1*m, hh - 0.5*dz) with 3 dofs;
54    add node # 15+18*e at (0.2*m, 0.1*m, hh - 0.5*dz) with 3 dofs;
55    add node # 16+18*e at (0.0*m, 0.2*m, hh - 0.5*dz) with 3 dofs;
56    add node # 17+18*e at (0.1*m, 0.2*m, hh - 0.5*dz) with 3 dofs;
57    add node # 18+18*e at (0.2*m, 0.2*m, hh - 0.5*dz) with 3 dofs;
58    add node # 19+18*e at (0.0*m, 0.0*m, hh) with 3 dofs;
59    add node # 20+18*e at (0.1*m, 0.0*m, hh) with 3 dofs;
60    add node # 21+18*e at (0.2*m, 0.0*m, hh) with 3 dofs;
61    add node # 22+18*e at (0.0*m, 0.1*m, hh) with 3 dofs;
62    add node # 23+18*e at (0.1*m, 0.1*m, hh) with 3 dofs;
63    add node # 24+18*e at (0.2*m, 0.1*m, hh) with 3 dofs;
64    add node # 25+18*e at (0.0*m, 0.2*m, hh) with 3 dofs;
65    add node # 26+18*e at (0.1*m, 0.2*m, hh) with 3 dofs;
66    add node # 27+18*e at (0.2*m, 0.2*m, hh) with 3 dofs;
67    add element # e+1 type 27NodeBrickLT with nodes
68    ( 21+18*e,
69     27+18*e,
70     25+18*e,
71     19+18*e,
72     3+18*e,
73     9+18*e,
74     7+18*e,
75     1+18*e,
76     24+18*e,
77     26+18*e,
22+18*e,
20+18*e,
6+18*e,
8+18*e,
4+18*e,
2+18*e,
12+18*e,
18+18*e,
16+18*e,
10+18*e,
14+18*e,
15+18*e,
17+18*e,
13+18*e,
11+18*e,
23+18*e,
5+18*e
)
use material # 1;
e += 1;
};
e = e -1;

// // --slowLoading-----------------------------------------------
// // add the 1 Newton load in 180 seconds.
// // ---------------------------------------------------------------------
// new loading stage "slowLoading";
// add load # 1 to node # (19+18*e) type path_time_series Fx=1/36.0*N ←
series_file = "slowLoading.txt";
// add load # 2 to node # (20+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";
// add load # 3 to node # (21+18*e) type path_time_series Fx=1/36.0*N ←
series_file = "slowLoading.txt";
// add load # 4 to node # (22+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";
// add load # 5 to node # (23+18*e) type path_time_series Fx=4/9.0*N ←
series_file = "slowLoading.txt";
// add load # 6 to node # (24+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";
// add load # 7 to node # (25+18*e) type path_time_series Fx=1/36.0*N ←
series_file = "slowLoading.txt";
// add load # 8 to node # (26+18*e) type path_time_series Fx=1/9.0*N ←
series_file = "slowLoading.txt";}
// add load # 9 to node # (27+18*e) type path_time_series Fx=1/36.0*N
    series_file = "slowLoading.txt";
// add algorithm and solver
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check;
// define solver ProfileSPD;
// simulate 2000 steps using transient algorithm
// time_step = 0.1*s;

// add the 1 Newton load in 0.6 seconds.

// new loading stage "fastLoading";
// add load # 101 to node # (19+18*e) type path_time_series Fx=1/36.0*N
    series_file = "fastLoading.txt";
// add load # 102 to node # (20+18*e) type path_time_series Fx=1/9.0*N
    series_file = "fastLoading.txt";
// add load # 103 to node # (21+18*e) type path_time_series Fx=1/36.0*N
    series_file = "fastLoading.txt";
// add load # 104 to node # (22+18*e) type path_time_series Fx=1/9.0*N
    series_file = "fastLoading.txt";
// add load # 105 to node # (23+18*e) type path_time_series Fx=4/9.0*N
    series_file = "fastLoading.txt";
// add load # 106 to node # (24+18*e) type path_time_series Fx=1/9.0*N
    series_file = "fastLoading.txt";
// add load # 107 to node # (25+18*e) type path_time_series Fx=1/36.0*N
    series_file = "fastLoading.txt";
// add load # 108 to node # (26+18*e) type path_time_series Fx=1/9.0*N
    series_file = "fastLoading.txt";
// add load # 109 to node # (27+18*e) type path_time_series Fx=1/36.0*N
    series_file = "fastLoading.txt";
// add algorithm and solver
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check;
// define solver ProfileSPD;
// simulate 1000 steps using transient algorithm
// time_step = 0.01*s;

// add a load and then release to free vibration

---freeVibration---
// add a load and then release to free vibration

new loading stage "freeVibration";
add load # 201 to node # (19+18*e) type path_time_series Fx=1/36.0*N
    series_file = "freeVibration.txt";
add load # 202 to node # (20+18*e) type path_time_series Fx=1/9.0*N
    series_file = "freeVibration.txt";
add load # 203 to node # (21+18*e) type path_time_series Fx=1/36.0*N
    series_file = "freeVibration.txt";
add load # 204 to node # (22+18*e) type path_time_series Fx=1/9.0*N series_file ← "freeVibration.txt";
add load # 205 to node # (23+18*e) type path_time_series Fx=4/9.0*N series_file ← "freeVibration.txt";
add load # 206 to node # (24+18*e) type path_time_series Fx=1/9.0*N series_file ← "freeVibration.txt";
add load # 207 to node # (25+18*e) type path_time_series Fx=1/36.0*N series_file = "freeVibration.txt";
add load # 208 to node # (26+18*e) type path_time_series Fx=1/9.0*N series_file ← "freeVibration.txt";
add load # 209 to node # (27+18*e) type path_time_series Fx=1/36.0*N series_file = "freeVibration.txt";
// add algorithm and solver
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check ;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.1*s;
// end
bye;

Displacement Results.

![Displacement Results Graph](image)

Figure 707.15: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded [here](#).
Figure 707.16: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.17: Free vibration condition, vertical displacements of the cantilever tip.
707.6 Elastic Beam Element under Dynamic Loading with concentrated mass

Problem description:

![Diagram of cantilever-mass model](image)

Figure 707.18: The cantilever-mass model.

**ESSI model fei/DSL file:**

```plaintext
model name "beam-mass_1element";

// add node
add node # 1 at (0.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at (1.0*m, 0.0*m, 0.0*m) with 6 dofs;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
possion_ratio=0.3;

// add elements
add element # 1 type beam_elastic with nodes (1,2)
cross_section = b*h
elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2/(1+possion_ratio)
torsion_Jx = 0.33*b*h^3
```

 Jeremić et al., UCD and LBNL version: 3. August, 2020, 19:54
bending_Iy = b*h^3/12
bending_Iz = b*h^3/12
mass_density = rho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

// add boundary condition
fix node # 1 dofs all;

// add mass
beamMass=rho*A*L;
add mass to node # 2
mx = beamMass
my = beamMass
mz = beamMass
Imx = 0*beamMass*L^2
Imy = 0*beamMass*L^2
Imz = 0*beamMass*L^2;

// add slowLoading
new loading stage "slowLoading";
add load # 1 to node # 2 type path_time_series
Fz = 1.*N
series_file = "slowLoading.txt";
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 2000 steps using transient algorithm
time_step = 0.1*s;

// add fastLoading
remove load # 1;
new loading stage "fastLoading";
add load # 2 to node # 2 type path_time_series
Fz = 1.*N
series_file = "fastLoading.txt";
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 1000 steps using transient algorithm
time_step = 0.01*s;

// remove freeVibration
remove load # 2;
new loading stage "freeVibration";
add load # 3 to node # 2 type path_time_series
Fz = 1.*N
series_file = "freeVibration.txt";
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 1000 steps using transient algorithm
time_step = 0.01*s;
bye;

Displacement results against time series

Figure 707.19: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.20: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.21: Free vibration condition, vertical displacements of the cantilever tip.
707.7 Elastic Beam, 27 Node Brick Model With Concentrated Mass

Problem description:

\[ F \]

\[ 0.2m \]

\[ 0.2m \]

\[ 0.2m \]

Figure 707.22: The cantilever-mass model.

**ESSI model fei/DSL file:**

```plaintext
model name "brick-mass_1element" ;

// Geometry: width and height
b=0.2*m;
h=0.2*m;

// Materials: properties
natural_period = 1*s;
natural_frequency = 2*pi/natural_period;
elastic_constant = 1e9*N/m^2;
I=b*h^3/12.0;
A=b*h;
L=1*m;
rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
posissoin_ratio=0.3;

add material # 1 type linear_elastic_isotropic_3d_LT
    mass_density = rho
    elastic_modulus = elastic_constant
    poisson_ratio = posissoin_ratio;

add node # 1 at ( 0.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 2 at ( 0.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 3 at ( 1.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 4 at ( 1.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
add node # 5 at ( 0.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 6 at ( 1.0000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
add node # 7 at ( 1.0000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
add node # 8 at ( 0.0000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
add node # 9 at ( 0.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
add node # 10 at ( 0.5000 *m, 0.2000 *m, 0.0000 *m) with 3 dofs;
```

Jeremić et al.  
UCD and LBNL  
version: 3. August, 2020, 19:54
32 add node # 11 at ( 1.0000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
33 add node # 12 at ( 0.5000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
34 add node # 13 at ( 0.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
35 add node # 14 at ( 0.5000 *m, 0.2000 *m, 0.2000 *m) with 3 dofs;
36 add node # 15 at ( 1.0000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
37 add node # 16 at ( 0.5000 *m, 0.0000 *m, 0.2000 *m) with 3 dofs;
38 add node # 17 at ( 0.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
39 add node # 18 at ( 0.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
40 add node # 19 at ( 1.0000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
41 add node # 20 at ( 1.0000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
42 add node # 21 at ( 0.5000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
43 add node # 22 at ( 0.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
44 add node # 23 at ( 0.5000 *m, 0.2000 *m, 0.1000 *m) with 3 dofs;
45 add node # 24 at ( 1.0000 *m, 0.1000 *m, 0.1000 *m) with 3 dofs;
46 add node # 25 at ( 0.5000 *m, 0.0000 *m, 0.1000 *m) with 3 dofs;
47 add node # 26 at ( 0.5000 *m, 0.1000 *m, 0.0000 *m) with 3 dofs;
48 add node # 27 at ( 0.5000 *m, 0.1000 *m, 0.2000 *m) with 3 dofs;
49 add element # 1 type 27NodeBrickLT with nodes( 2, 1, 3, 4, 5, 8, 7, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27) use material # 1;
50 fix node # 1 dofs all;
51 fix node # 2 dofs all;
52 fix node # 5 dofs all;
53 fix node # 8 dofs all;
54 fix node # 9 dofs all;
55 fix node # 13 dofs all;
56 fix node # 17 dofs all;
57 fix node # 18 dofs all;
58 fix node # 22 dofs all;
59 // Mapping from 3 dofs to 6 dofs.
60 add node # 1003 at ( 1.0000 *m, 0.2000 *m, 0.0000 *m) with 6 dofs;
61 add node # 1004 at ( 1.0000 *m, 0.0000 *m, 0.0000 *m) with 6 dofs;
62 add node # 1006 at ( 1.0000 *m, 0.0000 *m, 0.2000 *m) with 6 dofs;
63 add node # 1007 at ( 1.0000 *m, 0.2000 *m, 0.2000 *m) with 6 dofs;
64 // And connect the nodes at the same location.
65 add constraint equal dof with master node # 3 and slave node # 1003 dof to constrain ux uy uz;
66 add constraint equal dof with master node # 4 and slave node # 1004 dof to constrain ux uy uz;
67 add constraint equal dof with master node # 6 and slave node # 1006 dof to constrain ux uy uz;
68 add constraint equal dof with master node # 7 and slave node # 1007 dof to constrain ux uy uz;
69 add mass to node # 24 mx = rho*A*L my = rho*A*L mz = rho*A*L;
70 // add 6 beams to connect the mass
smallb=0.01*m;
smallh=0.01*m;
smallE = 1e9*N/m^2;
smallnu=0.3;
smallrho=0*kg/m^3;
smallI=smallb*smallh^3/12.0;
add element # 11 type beam_elastic with nodes (1003,1004)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
add element # 12 type beam_elastic with nodes (1003,1006)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
add element # 13 type beam_elastic with nodes (1003,1007)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
add element # 14 type beam_elastic with nodes (1004,1006)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = ( 1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
add element # 15 type beam_elastic with nodes (1004,1007)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);
add element # 16 type beam_elastic with nodes (1006,1007)
cross_section = smallb*smallh
elastic_modulus = smallE
shear_modulus = smallE/2/(1+smallnu)
torsion_Jx = 0.33*smallb*smallh^3
bending_Iy = smallI
bending_Iz = smallI
mass_density = smallrho
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

---

new loading stage "slowLoading";
add load # 1 to node # 4 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
add load # 2 to node # 6 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
add load # 3 to node # 3 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
add load # 4 to node # 7 type path_time_series Fz=1/36.0*N series_file = "slowLoading.txt";
add load # 5 to node # 20 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
add load # 6 to node # 11 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
add load # 7 to node # 15 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
add load # 8 to node # 19 type path_time_series Fz=1/9.0*N series_file = "slowLoading.txt";
add load # 9 to node # 24 type path_time_series Fz=4/9.0*N series_file = "slowLoading.txt";
add algorithm and solver
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 2000 steps using transient algorithm
// time_step = 0.1*s;
// // --fastLoading---------------------------------------------------------------
// add the 1 Newton load in 0.6 seconds.
// // new loading stage "fastLoading";
// add load # 101 to node # 4 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
// add load # 102 to node # 6 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
// add load # 103 to node # 3 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
// add load # 104 to node # 7 type path_time_series Fz=1/36.0*N series_file = "fastLoading.txt";
// add load # 105 to node # 20 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
// add load # 106 to node # 11 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
// add load # 107 to node # 15 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
// add load # 108 to node # 19 type path_time_series Fz=1/9.0*N series_file = "fastLoading.txt";
// add load # 109 to node # 24 type path_time_series Fz=4/9.0*N series_file = "fastLoading.txt";
// add algorithm and solver
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check;
// define solver ProfileSPD;
// simulate 1000 steps using transient algorithm
// time_step = 0.01*s;
// // --freeVibration---------------------------------------------------------------
// new loading stage "freeVibration";
// add load # 201 to node # 4 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
// add load # 202 to node # 6 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
// add load # 203 to node # 3 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
// add load # 204 to node # 7 type path_time_series Fz=1/36.0*N series_file = "freeVibration.txt";
// add load # 205 to node # 20 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
// add load # 206 to node # 11 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
// add load # 207 to node # 15 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
add load # 208 to node # 19 type path_time_series Fz=1/9.0*N series_file = "freeVibration.txt";
add load # 209 to node # 24 type path_time_series Fz=4/9.0*N series_file = "freeVibration.txt";
// add algorithm and solver
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
time_step = 0.1*s;

// end
bye;

---

Displacement Results.

![Displacement Graph](image)

Figure 707.23: Slow loading condition, vertical displacements of the cantilever tip.

The ESSI model fei/DSL files for this example can be downloaded [here](#).
Figure 707.24: Fast loading condition, vertical displacements of the cantilever tip.

Figure 707.25: Free vibration condition, vertical displacements of the cantilever tip.
707.8 Elastic Beam Element, Dynamic Loading, Viscous (Rayleigh/-Caughey) and Numerical (Newmark/HHT) Damping

Problem description:

Figure 707.26: The cantilever-mass model.

ESSI model fei/DSL file:

```plaintext
1 model name "beam_1element" ;
2 // add node
3 add node # 1 at ( 0.0*m , 0.0*m, 0.0*m) with 6 dofs;
4 add node # 2 at ( 1.0*m, 0.0*m, 0.0*m) with 6 dofs;
5 // Geometry: width and height
6 b=0.2*m;
7 h=0.2*m;
8 // Materials: properties
9 natural_period = 1*s;
10 natural_frequency = 2*pi/natural_period;
11 elastic_constant = 1e9*N/m^2;
12 I=b*h^3/12.0;
13 A=b*h;
14 L=1*m;
15 rho = (1.8751)^4*elastic_constant*I/(natural_frequency^2*L^4*A);
16 possion_ratio=0.3;
17 // add elements
18 add element # 1 type beam_elastic with nodes (1,2)
19 cross_section = b*h
20 elastic_modulus = elastic_constant
21 shear_modulus = elastic_constant/2/(1+possion_ratio)
22 torsion_Jx = 0.33*b*h^3
23 bending_Iy = b*h^3/12
24 bending_Iz = b*h^3/12
25 mass_density = rho
```
xz_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m)
joint_2_offset = (0*m, 0*m, 0*m);

// add boundary condition
fix node # 1 dofs all;

// // --no-damping-------------------------------------------------------------
// // "no-damping"
// new loading stage "no-damping";
// add load # 1 to node # 2 type path_time_series
// Fz = 1.*N
// series_file = "freeVibration.txt";
// define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
// define algorithm With_no_convergence_check;
// define solver ProfileSPD;
// simulate 100 steps using transient algorithm
// time_step = 0.1*s;

--Newmark-damping--

// // --HHT-damping-------------------------------------------------------------
// remove load # 3;
// new loading stage "HHT-damping";
// add load # 4 to node # 6 type path_time_series
// Fz = 1.*kN
// series_file = "freeVibration.txt";
// define dynamic integrator Hilber_Hughes_Taylor with alpha = -0.20;
// define algorithm With_no_convergence_check;
// define solver ProfileSPD;
// simulate 300 steps using transient algorithm
// time_step = 0.1*s;

--Rayleigh-damping--
// remove load # 4;
// simulate using eigen algorithm number_of_modes = 2;
f1=0.996807/s;
f2=0.996807/s;
w1 = 2*pi*f1;
w2 = 2*pi*f2;
xi=0.05;
rayl_a1 = 2*xi/(w1 + w2);
rayl_a0 = rayl_a1*w1*w2;
add damping # 1 type Rayleigh with
  a0 = rayl_a0
  a1 = rayl_a1
  stiffness_to_use = Initial_Stiffness;
add damping # 1 to element # 1;

new loading stage "Rayleigh-damping";
add load # 5 to node # 2 type path_time_series
  Fz = 1.*N
  series_file = "freeVibration.txt";
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 100 steps using transient algorithm
  time_step = 0.1*s;

// --Caughey3rd-damping-----------------------------------------------
// --Caughey3rd-damping-----------------------------------------------
// add damping # 2 type Caughey3rd with
//  a0 = 0.560523/s
//  a1 = 0.0730746*s
//  a2 = 0.000361559*s^3
//  stiffness_to_use = Last_Committed_Stiffness;
//  kk=1;
// while (kk<6) {
//  add damping # 2 to element # kk;
//  kk+=1;
// }
// new loading stage "Caughey3rd-damping";
// add load # 6 to node # 6 type path_time_series
//  Fz = 10.*kN
//  series_file = "freeVibration.txt";
// For Caughey3rd damping, we have to add some Newmark damping,
// Otherwise, there will be some high frequency noise.
// define dynamic integrator Newmark with gamma = 0.6 beta = 0.3025;
// define algorithm With_no_convergence_check;
// define solver ProfileSPD;
simulate 100 steps using transient algorithm
  time_step = 0.2*s;
Displacement results against time series

The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.27: Free vibration condition, no damping, vertical displacements of the cantilever tip.

Figure 707.28: Free vibration condition, viscous (Rayleigh) damping, vertical displacements of the cantilever tip.
Figure 707.29: Free vibration condition, viscous (Caughey3rd) damping, vertical displacements of the cantilever tip.

Figure 707.30: Free vibration condition, viscous (Caughey4th) damping, vertical displacements of the cantilever tip.
Figure 707.31: Free vibration condition, numerical (Newmark) damping, vertical displacements of the cantilever tip.

Figure 707.32: Free vibration condition, numerical (HHT) damping, vertical displacements of the cantilever tip.
707.9 Elastic Beam Element for a Simple Frame Structure

Problem Description

- Dimensions: width=6m, height=6m, force=100N

- Element dimensions: length=6m, cross section width=1m, cross section height=1m, mass density $\rho = 0.0 kN/m^3$, Young’s modulus $E = 1E8$ Pa, Poisson’s ratio $\nu = 0.0$.

![Elastic frame with beam elastic elements.](image)

ESSI model fei/DSL file:

```plaintext
model name "beam_element_presentation";
add node # 1 at ( 0.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 2 at ( 0.00*m, 0.00*m, 6.00*m) with 6 dofs;
add node # 3 at ( 6.00*m, 0.00*m, 6.00*m) with 6 dofs;
add node # 4 at ( 6.00*m, 0.00*m, 0.00*m) with 6 dofs;
elastic_constant = 1e8*N/m^2;
b=1*m;
h=1*m;
rho = 0*kg/m^3; // Mass density
add element # 1 type beam_elastic with nodes (1, 2)
cross_section = b*h elastic_modulus = elastic_constant
shear_modulus = elastic_constant/2
torsion_Jx = 0.33*b*h^3 bending_Iy = b*h^3/12 bending_Iz = h*b^3/12
mass_density = rho xx_plane_vector = (1, 0, 1)
joint_1_offset = (0*m, 0*m, 0*m) joint_2_offset = (0*m, 0*m, 0*m);
add element # 2 type beam_elastic with nodes (2,3)
```

Jeremić et al., UCD and LBNL
version: 3. August, 2020, 19:54
cross_section = b*h  
elastic_modulus = elastic_constant  
shear_modulus = elastic_constant/2  
torsion_Jx = 0.33*b*h^3  
bending_Iy = b*h^3/12  
bending_Iz = h*b^3/12  

mass_density = rho  
xz_plane_vector = (1, 0, 1)  
joint_1_offset = (0*m, 0*m, 0*m)  
joint_2_offset = (0*m, 0*m, 0*m);  

add element # 3 type beam_elastic with nodes (3,4)  
cross_section = b*h  
elastic_modulus = elastic_constant  
shear_modulus = elastic_constant/2  
torsion_Jx = 0.33*b*h^3  
bending_Iy = b*h^3/12  
bending_Iz = h*b^3/12  

mass_density = rho  
xz_plane_vector = (1, 0, 1)  
joint_1_offset = (0*m, 0*m, 0*m)  
joint_2_offset = (0*m, 0*m, 0*m);  

fix node #1 dofs all;  
fix node #4 dofs all;  

new loading stage "Fz";  

add load # 1 to node # 2 type linear Fz=50*N;  

define algorithm With_no_convergence_check;  
define solver ProfileSPD;  
define load factor increment 1;  
simulate 1 steps using static algorithm;  

bye;

The ESSI model fei/DSL files for this example can be downloaded [here](#).
707.10 27NodeBrick Cantilever Beam, Static Load

Problem description:

Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \). The force direction is shown in Figure (707.34).

Numerical model:

The 27NodeBrick elements for cantilever beams is shown in Figure (707.35):

ESSI model fei/DSL file:

```
1 model name "6meter_cantilever_27brick" ;
2 3 add material # 1 type linear_elastic_isotropic_3d
4 5 mass_density = 0*kg/m^3
6 7 elastic_modulus = 1e8*N/m^2
8 9 poisson_ratio = 0.0;
10 11 add node # 1 at ( 0.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
12 13 add node # 2 at ( 0.00 *m, 0.00 *m, 0.00 *m) with 3 dofs;
```
add node # 3 at (6.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
add node # 4 at (5.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
add node # 5 at (4.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
add node # 6 at (3.00 *m, 1.00 *m, 0.00 *m) with 3 dofs;
... 
add node #117 at (5.50 *m, 0.50 *m, 1.00 *m) with 3 dofs;
add element # 1 type 27NodeBrickLT with nodes(2, 10, 8, 1, 15, 17, 28, 23, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47) use material # 1;
add element # 2 type 27NodeBrickLT with nodes(10, 11, 7, 8, 17, 18, 27, 28, 48, 49, 50, 51, 52, 53, 34, 38, 54, 55, 56, 57, 58, 59, 43, 60, 61) use material # 1;
add element # 3 type 27NodeBrickLT with nodes(11, 12, 6, 7, 18, 19, 26, 27, 62, 63, 64, 49, 65, 66, 67, 52, 54, 68, 69, 55, 70, 71, 72, 73, 58, 74, 75) use material # 1;
add element # 4 type 27NodeBrickLT with nodes(12, 13, 5, 6, 19, 20, 25, 26, 76, 77, 78, 63, 79, 80, 81, 66, 68, 82, 83, 69, 84, 85, 86, 87, 72, 88, 89) use material # 1;
add element # 5 type 27NodeBrickLT with nodes(13, 14, 4, 5, 20, 21, 24, 25, 90, 91, 92, 77, 93, 94, 95, 80, 82, 96, 97, 83, 98, 99, 100, 101, 86, 102, 103) use material # 1;
add element # 6 type 27NodeBrickLT with nodes(14, 9, 3, 4, 21, 16, 22, 24, 104, 105, 106, 91, 107, 108, 109, 94, 96, 110, 111, 97, 112, 113, 114, 115, 100, 116, 117) use material # 1;
fix node # 1 dofs all;
fix node # 2 dofs all;
fix node # 15 dofs all;
fix node # 23 dofs all;
fix node # 32 dofs all;
fix node # 36 dofs all;
fix node # 37 dofs all;
fix node # 40 dofs all;
fix node # 45 dofs all;
new loading stage "Fz";
add load # 1 to node # 13 type linear Fz=2.777778*N;
add load # 2 to node # 24 type linear Fz=2.777778*N;
add load # 3 to node # 3 type linear Fz=2.777778*N;
add load # 4 to node # 34 type linear Fz=2.777778*N;
add load # 5 to node # 182 type linear Fz=11.111111*N;
add load # 6 to node # 177 type linear Fz=11.111111*N;
add load # 7 to node # 180 type linear Fz=11.111111*N;
add load # 8 to node # 183 type linear Fz=11.111111*N;
add load # 9 to node # 186 type linear Fz=44.444444*N;
define algorithm With_no_convergence_check;
define solver UMFPack;
define load factor increment 1;
simulate 1 steps using static algorithm;

bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.11 4NodeANDES Cantilever Beam, Force Perpendicular to Plane

Problem description:
Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \).

![Diagram of cantilever beam with force perpendicular to plane](image)

Figure 707.36: Cantilever beams

Numerical model:
For a force direction perpendicular to the plane, only the bending deformation is present.
The model is shown in Figure (707.37).

![Diagram of 4NodeANDES elements for cantilever beams under force perpendicular to plane](image)

Figure 707.37: 4NodeANDES elements for cantilever beams under force perpendicular to plane.

ESSI model fei/DSL file:

```plaintext
model name "6meter_cantilever_4NodeANDES" ;
add material # 1 type linear_elastic_isotropic_3d
  mass_density = 0*kg/m^3
  elastic_modulus = 1e8*N/m^2
  poisson_ratio = 0.0;
add node # 1 at ( 0.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 2 at ( 6.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 3 at ( 1.0*m, 0.0*m, 0.0*m) with 6 dofs;
```

add node # 4 at (2.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 5 at (3.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 6 at (4.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 7 at (5.0*m, 0.0*m, 0.0*m) with 6 dofs;
add node # 8 at (6.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 9 at (0.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 10 at (5.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 11 at (4.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 12 at (3.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 13 at (2.0*m, 1.0*m, 0.0*m) with 6 dofs;
add node # 14 at (1.0*m, 1.0*m, 0.0*m) with 6 dofs;

h = 1*m;
add element # 1 type 4NodeShell_ANDES with nodes (1,3,14,9) use material # 1
    thickness = h ;
add element # 2 type 4NodeShell_ANDES with nodes (3,4,13,14) use material # 1
    thickness = h ;
add element # 3 type 4NodeShell_ANDES with nodes (4,5,12,13) use material # 1
    thickness = h ;
add element # 4 type 4NodeShell_ANDES with nodes (5,6,11,12) use material # 1
    thickness = h ;
add element # 5 type 4NodeShell_ANDES with nodes (6,7,10,11) use material # 1
    thickness = h ;
add element # 6 type 4NodeShell_ANDES with nodes (7,2,8,10) use material # 1
    thickness = h ;

fix node # 1 dofs all ;
fix node # 9 dofs all ;

new loading stage "Fz";
add load # 1 to node # 8 type linear Fz=50*N;
add load # 2 to node # 2 type linear Fz=50*N;
define algorithm With_no_convergence_check ;
define solver ProfileSPD ;
define load factor increment 1 ;
simulate 1 steps using static algorithm ;
bye ;

The ESSI model fei/DSL files for this example can be downloaded here.
707.12 4NodeANDES Cantilever Beams, In-Plane Force

**Problem description:**

Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, \( \nu = 0.0 \).

![Diagram of cantilever beam with in-plane force](image)

Figure 707.38: Problem description for cantilever beams with in plane force

**Numerical model:**

The 4NodeANDES elements under in-plane force is shown in Figure (707.39).

![Diagram of 4NodeANDES elements for cantilever beams under in-plane force](image)

Figure 707.39: 4NodeANDES elements for cantilever beams under in-plane force

**ESSI model fei/DSL file:**

```plaintext
model name "6meter_cantilever_4NodeANDES" ;

add material # 1 type linear_elastic_isotropic_3d
mass_density = 0*kg/m^3
elastic_modulus = 1e8*N/m^2
poisson_ratio = 0.0;

add node # 1 at ( 0.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 2 at ( 6.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 3 at ( 1.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 4 at ( 2.00*m, 0.00*m, 0.00*m) with 6 dofs;
```

Jeremić et al., Real-ESSI Lecture Notes 707.12. 4NODEANDES CANTILEVER BEAMS, ... page: 2696 of 2919
add node # 5 at (3.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 6 at (4.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 7 at (5.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 8 at (6.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 9 at (0.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 10 at (5.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 11 at (4.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 12 at (3.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 13 at (2.00*m, 1.00*m, 0.00*m) with 6 dofs;
add node # 14 at (1.00*m, 1.00*m, 0.00*m) with 6 dofs;

h = 1*m;
add element # 1 type 4NodeShell_ANDES with nodes (1,3,14,9) use material # 1 ←
  thickness = h ;
add element # 2 type 4NodeShell_ANDES with nodes (3,4,13,14) use material # 1 ←
  thickness = h ;
add element # 3 type 4NodeShell_ANDES with nodes (4,5,12,13) use material # 1 ←
  thickness = h ;
add element # 4 type 4NodeShell_ANDES with nodes (5,6,11,12) use material # 1 ←
  thickness = h ;
add element # 5 type 4NodeShell_ANDES with nodes (6,7,10,11) use material # 1 ←
  thickness = h ;
add element # 6 type 4NodeShell_ANDES with nodes (7,2,8,10) use material # 1 ←
  thickness = h ;

fix node # 1 dofs all;
fix node # 9 dofs all;

new loading stage "Fy";
add load # 1 to node # 8 type linear Fy=50*N;
add load # 2 to node # 2 type linear Fy=50*N;

define algorithm With_no_convergence_check ;
define solver ProfileSPD;
define load factor increment 1;
simulate 1 steps using static algorithm;

bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.13 27NodeBrick Cantilever Beams, Dynamic Input

Problem description:

Length=20m, Width=1m, Height=1m, E=504MPa, \(\nu = 0.4\).

All degree of freedoms at the bottom nodes are fixed.

The load is a self weight with a dynamic displacement of supports.

![Problem description for one simple dynamic example](image1)

Numerical model:

The numerical model applied 27NodeBrick to simulate the 1C (1 component) motion.

![Numerical model for one simple dynamic example](image2)

ESSI model fei/DSL file:

```plaintext
1 model name "dynamic_example";
2 add material # 1 type linear_elastic_isotropic_3d_LT
3 mass_density = 2000*kg/m^3
```
elastic_modulus = 504000000.00*Pa
poisson_ratio = 0.4;

add node No 1 at (0*m, 0*m, 0*m) with 3 dofs;
add node No 2 at (0*m, 0.5*m, 0*m) with 3 dofs;
add node No 3 at (0*m, 1*m, 0*m) with 3 dofs;
add node No 4 at (0.5*m, 0*m, 0*m) with 3 dofs;
add node No 5 at (0.5*m, 0.5*m, 0*m) with 3 dofs;
add node No 6 at (0.5*m, 1*m, 0*m) with 3 dofs;
... ...
add node No 369 at (1*m, 1*m, 20*m) with 3 dofs;

add element # 1 type 27NodeBrickLT with nodes (27,21,19,25,9,3,1,7,24,20,22,6,2,4,8,18,12,10,16,14,15,11,13,17,23,5) use material # 1;
add element # 2 type 27NodeBrickLT with nodes (45,39,37,43,27,21,19,25,42,38,40,44,24,20,22,26,36,30,28,34,32,33,29,31,35,41,23) use material # 1;
add element # 3 type 27NodeBrickLT with nodes (63,57,55,61,45,39,37,43,60,56,58,62,42,38,40,44,54,48,46,52,50,51,47,49,53,59,41) use material # 1;
add element # 4 type 27NodeBrickLT with nodes (81,75,73,79,63,57,55,61,78,74,76,80,60,56,58,62,72,66,64,70,68,69,65,67,71,77,69) use material # 1;
add element # 5 type 27NodeBrickLT with nodes (99,93,91,97,81,75,73,79,96,92,94,98,78,74,76,80,90,84,82,88,86,87,83,85,89,95,77) use material # 1;
... ...

add acceleration field # 1 ax = 0*g ay = 0*g az = -1*g;
add load # 1 to element # 1 type self_weight use acceleration field # 1;
add load # 2 to element # 2 type self_weight use acceleration field # 1;
add load # 3 to element # 3 type self_weight use acceleration field # 1;
add load # 4 to element # 4 type self_weight use acceleration field # 1;
add load # 5 to element # 5 type self_weight use acceleration field # 1;
add load # 6 to element # 6 type self_weight use acceleration field # 1;
... ...
add load # 20 to element # 20 type self_weight use acceleration field # 1;

fix node No 1 dofs uy uz;
fix node No 2 dofs uy uz;
fix node No 3 dofs uy uz;
fix node No 4 dofs uy uz;
fix node No 5 dofs uy uz;
fix node No 6 dofs uy uz;
... fix node No 369 dofs uy uz;

zeta = 0.0166667;
fq1 = 3.75;
fq2 = 11.25;
omega1 = 2*pi*fq1;
omega2 = 2*pi*fq2;
zeta1 = zeta;
zeta2 = zeta;
alpha1 = 2*omega1*omega2*(zeta1*omega2-zeta2*omega1)/(omega2*omega2-omega1*omega1);
beta1 = 2* (zeta2*omega2-zeta1*omega1)/(omega2*omega2-omega1*omega1);
add damping # 1
type Rayleigh
with
  a0 = alpha1/s
  a1 = beta1*s
stiffness_to_use = Initial_Stiffness;

add damping # 1 to element # 1;
add damping # 1 to element # 2;
add damping # 1 to element # 3;
add damping # 1 to element # 4;
add damping # 1 to element # 5;
add damping # 1 to element # 6;
add damping # 1 to element # 7;
add damping # 1 to element # 8;
add damping # 1 to element # 9;
add damping # 1 to element # 10;
add damping # 1 to element # 11;
add damping # 1 to element # 12;
add damping # 1 to element # 13;
add damping # 1 to element # 14;
add damping # 1 to element # 15;
add damping # 1 to element # 16;
add damping # 1 to element # 17;
add damping # 1 to element # 18;
add damping # 1 to element # 19;
add damping # 1 to element # 20;
new loading stage "impose_motion";
add imposed motion # 1001 to node # 1 dof ux
displacement_scale_unit = 1*m displacement_file = "dis.txt"
velocity_scale_unit = 1*m/s velocity_file = "vel.txt"
acceleration_scale_unit = 1*m/s^2 acceleration_file = "acc.txt";
add imposed motion # 1002 to node # 2 dof ux
displacement_scale_unit = 1*m displacement_file = "dis.txt"
velocity_scale_unit = 1*m/s velocity_file = "vel.txt"
acceleration_scale_unit = 1*m/s^2 acceleration_file = "acc.txt";
add imposed motion # 1003 to node # 3 dof ux
displacement_scale_unit = 1*m displacement_file = "dis.txt"
velocity_scale_unit = 1*m/s velocity_file = "vel.txt"
acceleration_scale_unit = 1*m/s^2 acceleration_file = "acc.txt";
add imposed motion # 1009 to node # 9 dof ux
displacement_scale_unit = 1*m displacement_file = "dis.txt"
velocity_scale_unit = 1*m/s  velocity_file = "vel.txt"
acceleration_scale_unit = 1*m/s^2  acceleration_file = "acc.txt";

define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
define algorithm With_no_convergence_check;
define solver ProfileSPD;
simulate 50 steps using transient algorithm  time_step = 0.005*s;
bye;

The ESSI model fei/DSL files for this example can be downloaded [here](#).
707.14 4NodeANDES Square Plate, Four Edges Clamped

Problem description:

Length=20m, Width=20m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.3$.

The four edges are clamped.

The load is a self weight.

Figure 707.42: Square plate with four edges clamped

Numerical model:

The element side length is 1 meter.

ESSI model fei/DSL file:

```
model name "square_plate";

add material # 1 type linear_elastic_isotropic_3d
  mass_density = 1e2*kg/m^3 elastic_modulus = 1e8*N/m^2 poisson_ratio = 0.3;

add node # 1 at ( 0.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 2 at ( 20.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 3 at ( 1.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 4 at ( 2.00*m, 0.00*m, 0.00*m) with 6 dofs;
add node # 5 at ( 3.00*m, 0.00*m, 0.00*m) with 6 dofs;
```
add node 6 at (4.00*m, 0.00*m, 0.00*m) with 6 dofs;
...
add node 441 at (19.00*m, 19.00*m, 0.00*m) with 6 dofs;
h = 1*m;
add element 1 type 4NodeShell_ANDES with nodes(1, 3, 81, 80) use material #1 thickness=h;
add element 2 type 4NodeShell_ANDES with nodes(3, 4, 100, 81) use material #1 thickness=h;
add element 3 type 4NodeShell_ANDES with nodes(4, 5, 119, 100) use material #1 thickness=h;
add element 4 type 4NodeShell_ANDES with nodes(5, 6, 138, 119) use material #1 thickness=h;
add element 5 type 4NodeShell_ANDES with nodes(6, 7, 157, 138) use material #1 thickness=h;
add element 6 type 4NodeShell_ANDES with nodes(7, 8, 176, 157) use material #1 thickness=h;
add element 400 type 4NodeShell_ANDES with nodes(441, 41, 22, 43) use material #1 thickness=h;
fix node 1 dofs all;
fix node 2 dofs all;

Figure 707.43: 4NodeANDES edge clamped square plate with element side length 1m
```plaintext
fix node # 3 dofs all;
fix node # 4 dofs all;
fix node # 5 dofs all;
fix node # 6 dofs all;
...
...
fix node # 80 dofs all;

new loading stage "self_weight";
add acceleration field # 1 ax = 0*g ay = 0*g az = 1*m/s^2;
add load # 1 to element # 1 type self_weight use acceleration field # 1;
add load # 2 to element # 2 type self_weight use acceleration field # 1;
add load # 3 to element # 3 type self_weight use acceleration field # 1;
add load # 4 to element # 4 type self_weight use acceleration field # 1;
add load # 5 to element # 5 type self_weight use acceleration field # 1;
add load # 6 to element # 6 type self_weight use acceleration field # 1;
...
...
add load # 400 to element # 400 type self_weight use acceleration field # 1;

define algorithm With_no_convergence_check;
define solver ProfileSPD;
define load factor increment 1;
simulate 1 steps using static algorithm;

bye;
```

The ESSI model fei/DSL files for this example can be downloaded [here].
707.15 One Dimensional DRM Model

**Problem description:**

A simple 1D DRM model is shown in Fig. (707.44). The “DRM element”, “Exterior node” and “Boundary node” are required to be designated in the DRM HDF5 input. The format and script for the HDF5 input is available in DSL/input manual.

![Figure 707.44: 1D DRM model.](image)

**Numerical model:**

**ESSI model fei/DSL file:**

```plaintext
model name "DRM";

//Material for soil
add material # 1 type linear_elastic_isotropic_3d_LT
mass_density = 2000*kg/m^3
elastic_modulus = 1300*MPa
poisson_ratio = 0.3;

//Material for DRM layer
add material # 2 type linear_elastic_isotropic_3d_LT
mass_density = 2000*kg/m^3
elastic_modulus = 1300*MPa
poisson_ratio = 0.3;

//Material for exterior layer
add material # 3 type linear_elastic_isotropic_3d_LT
mass_density = 2000*kg/m^3
elastic_modulus = 1300*MPa
poisson_ratio = 0.3;
```

Jeremić et al. UCD and LBNL version: 3. August, 2020, 19:54
Figure 707.45: 1D DRM model.

```plaintext
21 add node # 1 at ( 0.00*m, 0.00*m, 0.00*m) with 3 dofs;
22 add node # 2 at ( 5.00*m, 0.00*m, 0.00*m) with 3 dofs;
23 add node # 3 at ( 5.00*m, 5.00*m, 0.00*m) with 3 dofs;
24 add node # 4 at ( 0.00*m, 5.00*m, 0.00*m) with 3 dofs;
25 add node # 5 at ( 5.00*m, 0.00*m, 50.00*m) with 3 dofs;
26 add node # 6 at ( 5.00*m, 0.00*m, 5.00*m) with 3 dofs;
27 ...
28 ...
29 add node # 52 at ( 0.00*m, 5.00*m, -5.00*m) with 3 dofs;
30
//
32 add element # 1 type 8NodeBrickLT with nodes( 1, 4, 3, 2, 24, 44, 34, 6) use ←
   material # 1;
33 add element # 2 type 8NodeBrickLT with nodes( 24, 44, 34, 6, 23, 43, 33, 7) use ←
   material # 1;
34 ...
35 add element # 12 type 8NodeBrickLT with nodes( 48, 47, 45, 46, 52, 51, 49, 50) ←
   use material # 3;
36
//
38 fix node # 1 dofs uy ;
39 fix node # 1 dofs uz ;
```
fix node # 2 dofs uy ;
fix node # 2 dofs uz ;
fix node # 3 dofs uy ;
fix node # 3 dofs uz ;
fix node # 4 dofs uy ;
fix node # 4 dofs uz ;
...
fix node # 51 dofs ux ;

new loading stage "1D";
add domain reduction method loading # 1
hdf5_file = "input.hdf5";

define algorithm With_no_convergence_check ;
define solver ProfileSPD;
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;
simulate 999 steps using transient algorithm time_step = 0.01*s;

bye;

The ESSI model fei/DSL files for this example can be downloaded here.
The same model for this example with 27NodeBrickLT can be downloaded here.

**Long 1D DRM model 1000:1**

To show the wave propagation explicitly, a long 1D model (1000:1) similar to the 1D DRM model above was made in this section.

The model description is same to Fig.(707.44) except this model use far more soil elements.
The general view is shown in Fig.(707.46) below.

There is still now outgoing waves at the exterior layers, which is shown in Fig(707.47).
The ESSI model fei/DSL files for this example can be downloaded here.
The results can also be seen in this animation.
Figure 707.46: Long 1D DRM model
Figure 707.47: Long 1D DRM model: exterior layer
707.16 Three Dimensional DRM Model

Problem description:

As shown in Fig.(707.48), the DRM layer is used to add the earthquake motion.

![Diagram of 3D Domain Reduction Method example](image)

Figure 707.48: The diagram for 3D Domain Reduction Method example.

Numerical result:

ESSI model fei/DSL file:

```plaintext
text
1 model name "DRM" ;
2
3 //Material for soil
4 add material # 1 type linear_elastic_isotropic_3d_LT
5  mass_density = 2000*kg/m^3
6  elastic_modulus = 1300*MPa
7  poisson_ratio = 0.3;
8
9 //Material for DRM layer
10 add material # 2 type linear_elastic_isotropic_3d_LT
11  mass_density = 2000*kg/m^3
12  elastic_modulus = 1300*MPa
13  poisson_ratio = 0.3;
```

---

Jeremić et al., UCD and LBNL

version: 3. August, 2020, 19:54
Figure 707.49: Diagram for the 3D DRM model.
use material # 1;  
add element # 2 type 8NodeBrickLT with nodes( 3, 41, 50, 4, 151, 603, 684, 160) ←  
use material # 1;  
...  
add element # 2352 type 8NodeBrickLT with nodes( 2925, 2924, 2922, 2923, 2921, ←  
    2920, 2918, 2919) use material # 3;  
/
//  
fix node # 1332 dofs all ;  
fix node # 1334 dofs all ;  
...  
...  
fix node # 2924 dofs all ;  
new loading stage "3D";  
add domain reduction method loading # 1  
    hdf5_file = "input.hdf5";  
define algorithm With_no_convergence_check ;  
define solver ProfileSPD;  
define dynamic integrator Newmark with gamma = 0.5 beta = 0.25;  
simulate 999 steps using transient algorithm time_step = 0.01*s;  
bye;  

The ESSI model fei/DSL files for this example can be downloaded [here](#).  
The same model for this example with 27NodeBrickLT can be downloaded [here](#).
707.17 ShearBeam Element, Pisano Material

Problem description:

In the element type "ShearBeamLT", only one Gauss point exists. ShearBeamLT element was used here to test the Pisano material model.

Vertical force $F_z$ was used to apply confinement to the element. Then, cyclic force $F_x$ is used to load point.

![ShearBeam element](image)

Figure 707.50: ShearBeam element.

Results

Resulting stress-strain relationship is shown in Fig. (707.51).

ESSI model fei/DSL file:

```plaintext
model name "pisanoLT";
add node # 1 at (0*m,0*m,0*m) with 3 dofs;
add node # 2 at (0*m,0*m,1*m) with 3 dofs;
fix node # 1 dofs all;
```

Figure 707.51: Shear stress-strain response.

```plaintext
fix node # 2 dofs uy;

add material # 1 type New_PisanoLT
mass_density = 2000*kg/m^3
elastic_modulus_1atm = 325*MPa poisson_ratio = 0.3
M_in = 1.4 kd_in = 0.0 xi_in = 0.0 h_in = 700 m_in = 0.7
initial_confining_stress = 0*kPa n_in = 0 a_in = 0.0 eplcum_cr_in = 1e-6;

add element # 1 type ShearBeamLT with nodes (1, 2) 
cross_section = 1*m^2 use material # 1;
new loading stage "confinement";
add load # 1 to node # 2 type linear Fz = -200*kN;
define load factor increment 0.01;
define algorithm With_no_convergence_check;
define solver UMFPack;
simulate 100 steps using static algorithm;
new loading stage "test01";
gamma_max = 3e-3;
add imposed motion # 2 to node # 2 dof ux
displacement_scale_unit = gamma_max*m displacement_file = "input_sine.txt"
velocity_scale_unit = gamma_max*m/s velocity_file = "input_sine.txt"
acceleration_scale_unit = gamma_max*m/s^2 acceleration_file = "input_sine.txt";
define load factor increment 0.0005;
```
```
34 define algorithm With_no_convergence_check;
35 define solver UMFPack;
36 simulate 2000 steps using static algorithm;
37
38 bye;
```

The ESSI model fei/DSL files for this example can be downloaded [here](#).
707.18 8NodeBrickLT Element, Drucker-Prager Material, Armstrong-Frederick Rotational Kinematic Hardening

Problem description:

This example is used to test the materials properties, such as $G/G_{max}$ against strains. The element type is 8NodeBrickLT. And there are two stages of loading. The first loading stage is confinement and the second loading stage is shearing.

The boundary condition is specially designed such that each Gauss point has the same stress state.

Results

Resulting stress-strain relationship is shown in Fig.(707.52).

![Shear stress-strain response](image)

**Figure 707.52: Shear stress-strain response.**

**ESSI model fei/DSL file:**

```plaintext
// Drucker Prager Armstrong Frederick
// This model is created by Jose.
model name "druckeraf";

// Parameters:
phi = 5;
ha = 1000;
cr = 973;
```
gam = 0.01;
Ncyc = 5;
Nsteps = 1000;
H = 1;
vp = 1000*m/s;
vs = 500*m/s;
rho = 2000*kg/m^3;
p0 = 250*kPa;
G = rho*vs^2;
M = rho*vp^2;
E = G*(3*M-4*G)/(M-G);
nu = (M-2*G)/(2*M-2*G);
K0 = 1.0;
phirad = pi*phi/180;
M = 6*sin(phirad)/(3-sin(phirad));

// Define the material:
add material # 1 type DruckerPragerArmstrongFrederickLT
  mass_density = 0*kg/m^3
  elastic_modulus = E
  poisson_ratio = nu
  druckerprager_k = M
  armstrong_frederick_ha = ha*Pa
  armstrong_frederick_cr = cr*Pa
  isotropic_hardening_rate = 0*E
  initial_confining_stress = 1*Pa;

// define the node:
add node # 1 at (0*m,0*m,1*m) with 3 dofs;
add node # 2 at (1*m,0*m,1*m) with 3 dofs;
add node # 3 at (1*m,1*m,1*m) with 3 dofs;
add node # 4 at (0*m,1*m,1*m) with 3 dofs;
add node # 5 at (0*m,0*m,0*m) with 3 dofs;
add node # 6 at (1*m,0*m,0*m) with 3 dofs;
add node # 7 at (1*m,1*m,0*m) with 3 dofs;
add node # 8 at (0*m,1*m,0*m) with 3 dofs;

// add equal degree of freedom in three directions
add constraint equal dof with master node # 2 and slave node # 3 dof to constrain ux;
add constraint equal dof with master node # 2 and slave node # 6 dof to constrain ux;
add constraint equal dof with master node # 2 and slave node # 7 dof to constrain ux;
add constraint equal dof with master node # 3 and slave node # 4 dof to constrain uy;
add constraint equal dof with master node # 3 and slave node # 8 dof to
```plaintext
constrain uy;
add constraint equal dof with master node # 3 and slave node # 7 dof to constrain uy;
add constraint equal dof with master node # 1 and slave node # 2 dof to constrain uz;
add constraint equal dof with master node # 1 and slave node # 3 dof to constrain uz;
add constraint equal dof with master node # 1 and slave node # 4 dof to constrain uz;

// Define the element.
add element # 1 type 8NodeBrickLT with nodes (1, 2,3 , 4, 5, 6,7, 8) use material # 1;
new loading stage "confinement";
fix node # 1 dofs ux uy;
fix node # 2 dofs uy;
fix node # 4 dofs ux;
fix node # 5 dofs ux uy uz;
fix node # 6 dofs uy uz;
fix node # 7 dofs uz;
fix node # 8 dofs ux uz;

sigma_z = -3*p0/(1+2*K0);
sigma_x = K0*sigma_z;
sigma_y = K0*sigma_z;

//Z-face
add load # 1 to node # 1 type linear Fz = sigma_z*m^2/4;
add load # 2 to node # 2 type linear Fz = sigma_z*m^2/4;
add load # 3 to node # 3 type linear Fz = sigma_z*m^2/4;
add load # 4 to node # 4 type linear Fz = sigma_z*m^2/4;

//X-face
add load # 5 to node # 2 type linear Fx = sigma_x*m^2/4;
add load # 6 to node # 6 type linear Fx = sigma_x*m^2/4;
add load # 7 to node # 7 type linear Fx = sigma_x*m^2/4;
add load # 8 to node # 3 type linear Fx = sigma_x*m^2/4;
add load # 9 to node # 3 type linear Fy = sigma_y*m^2/4;
add load # 10 to node # 7 type linear Fy = sigma_y*m^2/4;
add load # 11 to node # 8 type linear Fy = sigma_y*m^2/4;
add load # 12 to node # 4 type linear Fy = sigma_y*m^2/4;

Nsteps_static=100;
define load factor increment 1/Nsteps_static;
define solver UMFPack;
define convergence test Norm_Displacement_Increment
```
tolerance = 1e-6
maximum_iterations = 100
verbose_level = 4;
define algorithm Newton;
define NDMaterialLT constitutive integration algorithm Euler_One_Step
yield_function_relative_tolerance = 0.002
stress_relative_tolerance = 0.002
maximum_iterations = 1000;
simulate Nsteps_static steps using static algorithm;

new loading stage "shearing";
compute reaction forces;
add load # 13 to node # 1 type from_reactions;
add load # 14 to node # 4 type from_reactions;
free node # 1 dofs ux;
free node # 4 dofs ux;
fix node # 3 dofs uy;
fix node # 6 dofs ux;
fix node # 7 dofs ux uy;
fix node # 8 dofs uy;
add constraint equal dof with master node # 1 and slave node # 3 dof to constrain ux;
add constraint equal dof with master node # 1 and slave node # 4 dof to constrain ux;
add constraint equal dof with master node # 1 and slave node # 2 dof to constrain ux;
remove constraint equaldof node # 6;
remove constraint equaldof node # 7;
remove constraint equaldof node # 8;
n = 1;
while(n<=1)
{
    add load # 14+n to node # n type path_time_series
    Fx = 170.*kN
    series_file = "path.txt";
    n+=1;
}
define load factor increment 1/Nsteps;
define solver UMFPack;
define convergence test Norm_Displacement_Increment
tolerance = 1e-5
maximum_iterations = 100
verbose_level = 4;
define algorithm Newton;

define NDMaterialLT constitutive integration algorithm Euler_One_Step
    yield_function_relative_tolerance = 0.0002
    stress_relative_tolerance = 0.002
    maximum_iterations = 1000;

simulate Ncyc*Nsteps steps using static algorithm;

bye;

The ESSI model fei/DSL files for this example can be downloaded here.
707.19 Contact Element Under Static Loading

Two Bar Normal Contact Problem Under Monotonic Loading.

This is an example of normal monotonic loading on a 1-D contact/interface between two bars separated by an initial gap of 0.1 unit. An illustrative diagram of the problem statement is shown below.

Figure 707.53: Illustration of Two Bar Normal Contact Problem under monotonic loading with initial gap

ESSI model fei/DSL file:

```plaintext
model name "Two_Bar_Contact_Under_Normal_Monotonic>Loading";

// Adding material
add material #1 type uniaxial_elastic elastic_modulus = 1*Pa ← viscoelastic_modulus = 0*Pa*s;

// Adding Nodes
add node #1 at (0*m,0*m,0*m) with 3 dofs;
add node #2 at (1*m,0*m,0*m) with 3 dofs;
add node #3 at (1.1*m,0*m,0*m) with 3 dofs;
add node #4 at (2.1*m,0*m,0*m) with 3 dofs;

// Adding Fixities
fix node #1 dofs ux uy uz;
fix node #4 dofs ux uy uz;
fix node #2 dofs uy uz;
fix node #3 dofs uy uz;

// Adding Truss Elements
add element #1 type truss with nodes (1,2) use material # 1 cross_section = ← 1*m^2 mass_density = 1*kg/m^3;
add element #2 type truss with nodes (3,4) use material # 1 cross_section = ← 1*m^2 mass_density = 1*kg/m^3;

// Adding Contact Element
add element #3 type FrictionalPenaltyContact with nodes (2,3)
normal_stiffness = 1e10*N/m
tangential_stiffness = 1e10*Pa*m
normal_damping = 0*kN/m*s
tangential_damping = 0*kN/m*s
friction_ratio = 0.3
```

Jeremić et al., Real-ESSI UCD and LBNL version: 3. August, 2020, 19:54
contact_plane_vector = (1,0,0);

new loading stage "Adding_Normal_Load";

add load #1 to node #2 type linear Fx = 0.3*N;

Nsteps = 10;

tol = 5e-12;

define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 10
verbose_level = 4;

define algorithm Newton;
define solver UMFPack;

define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

bye;

The displacement output of Node 2 and Node 3 are shown below.

![Figure 707.54: Displacement of Nodes 2 and 3](image)

The ESSI model fei/DSL files for this example can be downloaded [here](#).
707.20 Four Bar Contact Problem With Normal and Shear Force Under Monotonic Loading

This is an example to show the normal and tangential behaviour (stick and slip case) of contacts/interfaces using four bars in 2-D plane. The bars in x-directions are in contact (initial gap=0).

Figure 707.55: Illustration of Four Bar Normal Contact Problem With Normal and Shear Force Under Monotonic Loading with no initial gap

ESSI model fei/DSL file:
model name "Four_Bar_Contact_Under_Monotonic_Normal_and_Shear_Loading";

// Adding material
add material #1 type uniaxial_elastic elastic_modulus = 1*Pa ←
  viscoelastic_modulus = 0*Pa*s;

// Adding Nodes
add node #1 at (0*m,0*m,0*m) with 3 dofs;
add node #2 at (1*m,0*m,0*m) with 3 dofs;
add node #3 at (1*m,0*m,0*m) with 3 dofs;
add node #4 at (2*m,0*m,0*m) with 3 dofs;
add node #5 at (1*m,-1*m,0*m) with 3 dofs;
add node #6 at (1*m,1*m,0*m) with 3 dofs;

// Adding Truss Elements
add element #1 type truss with nodes (1,2) use material # 1 cross_section = ←
  1*m^2 mass_density = 1*kg/m^3;
add element #2 type truss with nodes (3,4) use material # 1 cross_section = ←
  1*m^2 mass_density = 1*kg/m^3;
add element #3 type truss with nodes (3,5) use material # 1 cross_section = ←
  1*m^2 mass_density = 1*kg/m^3;
add element #4 type truss with nodes (2,6) use material # 1 cross_section = ←
  1*m^2 mass_density = 1*kg/m^3;

// Adding Contact Element
add element #5 type FrictionalPenaltyContact with nodes (2,3)
  normal_stiffness = 1e12*N/m
  tangential_stiffness = 1e12*N/m
  normal_damping = 0*N/m*s
  tangential_damping = 0*N/m*s
  friction_ratio = 0.4
  contact_plane_vector = (1,0,0);

// Adding Fixities
fix node #1 dofs ux uy uz ;
fix node #4 dofs ux uy uz ;
fix node #5 dofs ux uy uz ;
fix node #6 dofs ux uy uz ;
fix node #2 dofs uz ;
fix node #3 dofs uz ;

new loading stage "Normal_Loading";

add load #1 to node #2 type linear Fx = 0.1*N;

tol = 1e-10;
define convergence test Norm_Displacement_Increment
  tolerance = tol
  maximum_iterations = 10
  verbose_level = 4;
```plaintext
define algorithm Newton;

Nsteps = 10;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

new loading stage "Shear_Loading";

add load #2 to node #2 type linear Fy = 0.2*N;
tol = 1e-10;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 10
verbose_level = 4;

define algorithm Newton;

Nsteps = 100;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;
bye;
```

The displacement output of Node 2 and Node 3 are shown below.
The ESSI model fei/DSL files for this example can be downloaded here.
Figure 707.56: Displacement of Nodes 2 and 3 along y direction
707.21 3-D Truss example with normal confinement and Shear Loading

A simple 3-D truss example with Normal confinement in z-direction of $F_N = 0.5N$, friction coefficient $\mu = 0.2$ and shear loading of magnitude $F_s = 0.5N$. Figure 707.57 below, shows the description of the problem.

Figure 707.57: Illustration of 3-D Truss Problem with confinement loading in z-direction of 0.5N and then shear loading of 0.5N in x-y plane

ESSI model fei/DSL file:

```plaintext
model name "3-D Contact Under Normal And Tangential Loading" ;

// Adding material
add material #1 type uniaxial_elastic elastic_modulus = 1*Pa ← viscoelastic_modulus = 0*Pa*s;

// Adding Nodes
add node #1 at (0*m,0*m,0*m) with 3 dofs;
add node #2 at (0*m,0*m,0*m) with 3 dofs;
add node #3 at (-1*m,0*m,0*m) with 3 dofs;
add node #4 at (0*m,1*m,0*m) with 3 dofs;
add node #5 at (0*m,0*m,1*m) with 3 dofs;

// Adding Fixities
fix node #1 dofs ux uy uz;
fix node #3 dofs ux uy uz;
fix node #4 dofs ux uy uz;
fix node #5 dofs ux uy uz;

// Adding Truss Elements
add element #1 type truss with nodes (2,3) use material # 1 cross_section = ← 1*m^2 mass_density = 1*kg/m^3;
add element #2 type truss with nodes (2,4) use material # 1 cross_section = ←
```
1*m^2 mass_density = 1*kg/m^3;
add element #3 type truss with nodes (2,5) use material # 1 cross_section = 1*m^2 mass_density = 1*kg/m^3;

// Adding Contact Element
add element #4 type FrictionalPenaltyContact with nodes (1,2)
normal_stiffness = 1e10*N/m
tangential_stiffness = 1e10*Pa*m
normal_damping = 0*kN/m*s
tangential_damping = 0*kN/m*s
friction_ratio = 0.2
contact_plane_vector = (0,0,1);

new loading stage "Adding_Normal_Load";
add load #1 to node #2 type linear Fz = -0.5*N;
Nsteps = 1;
tol = 1e-10;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 1
verbose_level = 4;
define algorithm Newton;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

new loading stage "Shear_Loading";
add load #2 to node #2 type linear Fx = 0.4;
add load #3 to node #2 type linear Fy = 0.3;
tol = 1e-12;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 10
verbose_level = 4;
define algorithm Newton;
Nsteps = 20;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;
bye;
The generalized displacement response of the tangential loading stage is shown below.

![Chart showing displacements](image1)

![Chart showing forces](image2)

Figure 707.58: Displacements of Node 2 with applied shear tangential load step.

Figure 707.59: Resisting force by the contact/interface element with applied shear tangential load step.

The ESSI model fei/DSL files for this example can be downloaded [here](#).
707.22 Six Solid Blocks Example With Contact

This is a 3-D solid block example with initial normal and then tangential load on different surfaces as shown below.

Figure 707.60: Illustration of Six Solid Blocks Example with Contact having first normal and then tangential loading stages.

ESSI model fei/DSL file:

```plaintext
model name "Six_Solid_Blocks_Example_With_Contact";

// Adding material
add material #1 type linear_elastic_isotropic_3d_LT mass_density=2000*kg/m^3 ←
elastic_modulus=200*MPa poisson_ratio=0.3;

// Adding Nodes
add node # 1 at (-1.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 2 at (-1.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 3 at (1.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 4 at (1.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 5 at (-1.500000*m,-0.500000*m,2.000000*m) with 3 dofs;
add node # 6 at (-1.500000*m,0.500000*m,2.000000*m) with 3 dofs;
add node # 7 at (0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 8 at (0.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 9 at (-0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 10 at (0.500000*m,-0.500000*m,2.000000*m) with 3 dofs;
add node # 11 at (-0.500000*m,0.500000*m,2.000000*m) with 3 dofs;
add node # 12 at (0.500000*m,0.500000*m,2.000000*m) with 3 dofs;
add node # 13 at (-0.500000*m,0.500000*m,-2.000000*m) with 3 dofs;
```

Jeremić, J., et al. Real-ESSI Lecture Notes 707.22. SIX SOLID BLOCKS EXAMPLE WITH... page: 2730 of 2919
add node # 14 at (0.500000*m,0.500000*m,-2.000000*m) with 3 dofs;
add node # 15 at (0.500000*m,-0.500000*m,-2.000000*m) with 3 dofs;
add node # 16 at (-0.500000*m,-0.500000*m,-2.000000*m) with 3 dofs;
add node # 17 at (-1.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 18 at (-1.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 19 at (1.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 20 at (1.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 21 at (-0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 22 at (0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 23 at (-0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 24 at (0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 25 at (-0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 26 at (0.500000*m,-0.500000*m,0.000000*m) with 3 dofs;
add node # 27 at (-0.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 28 at (0.500000*m,0.500000*m,0.000000*m) with 3 dofs;
add node # 29 at (-0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 30 at (0.500000*m,0.500000*m,-1.000000*m) with 3 dofs;
add node # 31 at (-0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;
add node # 32 at (0.500000*m,-0.500000*m,-1.000000*m) with 3 dofs;

// Adding Solid 8 Node Brick Elements
add element #1 type 8NodeBrickLT with nodes (21,23,17,18,11,9,1,2) use ←
   material #1;
add element #2 type 8NodeBrickLT with nodes (13,16,5,6,21,23,17,18) use ←
   material #1;
add element #3 type 8NodeBrickLT with nodes (30,32,31,29,28,26,25,27) use ←
   material #1;
add element #4 type 8NodeBrickLT with nodes (14,15,16,13,22,24,23,21) use ←
   material #1;
add element #5 type 8NodeBrickLT with nodes (19,20,24,22,4,3,10,12) use ←
   material #1;
add element #6 type 8NodeBrickLT with nodes (7,8,15,14,19,20,24,22) use ←
   material #1;

// Adding some variables
Kn = 1e12*N/m; // normal penalty stiffness
Kt = 1e12*N/m; // tangential penalty stiffness
Cn = 0*N/m*s; // normal penalty damping
Ct = 0*N/m*s; // tangential penalty damping
nu = 0.4; // friction ratio

// Adding Contact Element
add element #7 type FrictionalPenaltyContact with nodes (9,25)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (1,0,0);
add element #8 type FrictionalPenaltyContact with nodes (10,26)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (-1,0,0);

add element #9 type FrictionalPenaltyContact with nodes (11,27)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (1,0,0);

add element #10 type FrictionalPenaltyContact with nodes (12,28)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (-1,0,0);

add element #11 type FrictionalPenaltyContact with nodes (21,29)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (1,0,0);

add element #12 type FrictionalPenaltyContact with nodes (22,30)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (-1,0,0);

add element #13 type FrictionalPenaltyContact with nodes (23,31)
  normal_stiffness = Kn
  tangential_stiffness = Kt
  normal_damping = Cn
  tangential_damping = Ct
  friction_ratio = nu
  contact_plane_vector = (1,0,0);

add element #14 type FrictionalPenaltyContact with nodes (24,32)
  normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (-1,0,0);

add element #15 type FrictionalPenaltyContact with nodes (21,29)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

add element #16 type FrictionalPenaltyContact with nodes (22,30)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

add element #17 type FrictionalPenaltyContact with nodes (23,31)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

add element #18 type FrictionalPenaltyContact with nodes (24,32)
normal_stiffness = Kn
tangential_stiffness = Kt
normal_damping = Cn
tangential_damping = Ct
friction_ratio = nu
contact_plane_vector = (0,0,1);

// Adding Fixities
fix node #5 dofs ux uy uz;
fix node #6 dofs ux uy uz;
fix node #13 dofs ux uy uz;
fix node #16 dofs ux uy uz;
fix node #15 dofs ux uy uz;
fix node #14 dofs ux uy uz;
fix node #7 dofs ux uy uz;
fix node #8 dofs ux uy uz;
fix node #17 dofs ux uy;
fix node #18 dofs ux uy;
fix node #1 dofs ux uy;
fix node #2 dofs ux uy;
fix node #20 dofs ux uy;
fix node #19 dofs ux uy;
fix node #3 dofs ux uy;
fix node #4 dofs ux uy;
fix node #9 dofs uy;
fix node #10 dofs uy;
fix node #23 dofs uy;
fix node #24 dofs uy;
fix node #11 dofs uy;
fix node #21 dofs uy;
fix node #12 dofs uy;
fix node #22 dofs uy;
fix node #25 dofs uy;
fix node #26 dofs uy;
fix node #27 dofs uy;
fix node #28 dofs uy;
fix node #29 dofs uy;
fix node #30 dofs uy;
fix node #31 dofs uy;
fix node #32 dofs uy;

new loading stage "Normal_Loading";

add load #1 to element #3 type surface at nodes (25,26,27,28) with magnitude → (-1*Pa);

tol = 1e-12;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 100
verbose_level = 4;

define algorithm Newton;

Nsteps = 10;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

new loading stage "Shear_Loading";

add load #2 to element #3 type surface at nodes (26,28,30,32) with magnitude ← (-1*Pa);

tol = 1e-12;
define convergence test Norm_Displacement_Increment
tolerance = tol
maximum_iterations = 100
verbose_level = 4;

define algorithm Newton;
Nsteps = 10;
define solver UMFPack;
define load factor increment 1/Nsteps;
simulate Nsteps steps using static algorithm;

bye;

The generalized displacement field of the two loading stages normal loading and tangential loading is shown below.

Figure 707.61: Generalized displacement magnitude visualization of normal loading

Figure 707.62: Generalized displacement magnitude visualization of tangential loading
The ESSI model fei/DSL files for this example can be downloaded here.
707.23 Pure shear model for $G/G_{\text{max}}$ plot

Problem description:

The pure shear model for $G/G_{\text{max}}$ plot

Figure 707.63: The pure shear model for (a) confinement and (b) shearing

ESSI model fei/DSL file:

```plaintext
model name "GGmax" ;
// Parameters:
phi = 0.0135713590083;
ha = 2.94767923453;
fr = 1854.31984573;
rho=1922.5;
depth=0.1524/2;
confinstress=9.8*depth*rho;
G=12388.33;
p0 = confinstress*Pa;
phirad = pi*phi/180;
M = 6*sin(phirad)/(3-sin(phirad));
nu=0.3;
add material # 1 type DruckerPragerArmstrongFrederickLT
mass_density = rho*kg/m^3
elastic_modulus = 2*G*(1+nu)*Pa
poisson_ratio = nu
druckerprager_k = M
druckerprager_ha = ha*Pa
armstrong_frederick_ha = ha*Pa
armstrong_frederick_cr = cr*Pa
```
```
  isotropic_hardening_rate = 0*Pa
  initial_confining_stress = 10*Pa;
  add node # 1 at ( 1.0000 *m, 0.0000 *m, 0.0000 *m) with 3 dofs;
  add node # 2 at ( 0.0000 *m, 1.0000 *m, 0.0000 *m) with 3 dofs;
  add node # 3 at ( 1.0000 *m, 2.0000 *m, 0.0000 *m) with 3 dofs;
  add node # 4 at ( 2.0000 *m, 1.0000 *m, 0.0000 *m) with 3 dofs;
  add node # 5 at ( 1.0000 *m, 0.0000 *m, 1.0000 *m) with 3 dofs;
  add node # 6 at ( 0.0000 *m, 1.0000 *m, 1.0000 *m) with 3 dofs;
  add node # 7 at ( 1.0000 *m, 2.0000 *m, 1.0000 *m) with 3 dofs;
  add node # 8 at ( 2.0000 *m, 1.0000 *m, 1.0000 *m) with 3 dofs;
  add element # 1 type 8NodeBrickLT with nodes(1,2,3,4,5,6,7,8) use material # 1;

  // fix the y direction for node 2,4,6,8
  fix node # 2 dofs uy;
  fix node # 4 dofs uy;
  fix node # 6 dofs uy;
  fix node # 8 dofs uy;

  // fix the x direction for node 1,3,5,7
  fix node # 1 dofs ux;
  fix node # 3 dofs ux;
  fix node # 5 dofs ux;
  fix node # 7 dofs ux;

  // Stage 1: confinement
  new loading stage "confinement";
  add load # 1 to node # 1 type linear Fy= p0*m^2;
  add load # 2 to node # 3 type linear Fy= -p0*m^2;
  add load # 3 to node # 5 type linear Fy= p0*m^2;
  add load # 4 to node # 7 type linear Fy= -p0*m^2;

  add load # 5 to node # 2 type linear Fx= p0*m^2;
  add load # 6 to node # 4 type linear Fx= -p0*m^2;
  add load # 7 to node # 6 type linear Fx= p0*m^2;
  add load # 8 to node # 8 type linear Fx= -p0*m^2;

  // confinement at z direction
  add load # 101 to node # 1 type linear Fz= p0*m^2;
  add load # 102 to node # 2 type linear Fz= p0*m^2;
  add load # 103 to node # 3 type linear Fz= p0*m^2;
  add load # 104 to node # 4 type linear Fz= p0*m^2;

  add load # 105 to node # 5 type linear Fz= -p0*m^2;
  add load # 106 to node # 6 type linear Fz= -p0*m^2;
  add load # 107 to node # 7 type linear Fz= -p0*m^2;
  add load # 108 to node # 8 type linear Fz= -p0*m^2;

  // add algorithm and solver
  Nsteps=100;
  define load factor increment 1/Nsteps;
  define solver ProfileSPD;
  define convergence test Norm_Disp_Increment
  tolerance = 1e-5
```
maximum_iterations = 100
verbose_level = 4;
// define algorithm With_no_convergence_check;
define algorithm Newton;
define NDMaterialLT constitutive integration algorithm Euler_One_Step
yield_function_relative_tolerance = 0.00002
stress_relative_tolerance = 0.0002
maximum_iterations = 1000;
simulate Nsteps steps using static algorithm;
// ------------------------------------------------------------------------
// Stage 2: shear
new loading stage "shear";
// fix all the uz, since we want plane strain.
i=1;
while (i<9) {
   remove load # 100+i;
   fix node # i dofs uz;
i=i+1;
}
shearforce=1.6*kN;

add load # 9 to node # 1 type linear Fy= shearforce;// series_file = "path.txt";
add load # 10 to node # 3 type linear Fy=-shearforce;// series_file = "path.txt"
add load # 11 to node # 5 type linear Fy= shearforce;// series_file = "path.txt"
add load # 12 to node # 7 type linear Fy=-shearforce;// series_file = "path.txt"
add load # 13 to node # 2 type linear Fx=-shearforce;// series_file = "path.txt"
add load # 14 to node # 4 type linear Fx= shearforce;// series_file = "path.txt"
add load # 15 to node # 6 type linear Fx=-shearforce;// series_file = "path.txt"
add load # 16 to node # 8 type linear Fx= shearforce;// series_file = "path.txt"

// add algorithm and solver
Nsteps=1e4;
define static integrator displacement_control using node # 1 dof uy increment ←
1e-2/Nsteps*m;
define convergence test Norm_Displacement_Increment tolerance = 0.000001 ←
maximum_iterations = 100 verbose_level = 0;
define solver ProfileSPD;
define algorithm Newton;
define NDMaterialLT constitutive integration algorithm Euler_One_Step
yield_function_relative_tolerance = 0.00002
stress_relative_tolerance = 0.0002
maximum_iterations = 1000;
simulate Nsteps steps using static algorithm;
bye;

Figure 707.64: The G/Gmax results

The ESSI model fei/DSL files for this example can be downloaded here.
707.24 Multi-yield-surface von-Mises for G/Gmax plot

Problem description:

This model illustrates the G/Gmax input to multi-yield-surface von-Mises material. This example is based on one Gauss-point with multi-yield-surface von-Mises material. The G/Gmax is converted to material modeling parameters (yield-surface size and hardening parameter) inside the DSL.

ESSI model fei/DSL file:

```plaintext
model name "GGmax";
add material # 1 type vonMisesMultipleYieldSurfaceGoverGmax
mass_density = 0.0*kg/m^3
initial_shear_modulus = 3E8 * Pa
poisson_ratio = 0.0
total_number_of_shear_modulus = 9
GoverGmax = "1,0.995,0.966,0.873,0.787,0.467,0.320,0.109,0.063"
ShearStrainGamma = "0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01"
incr_size = 0.000001 ;
max_strain= 0.005 ;
num_of_increm = max_strain/incr_size -1 ;
simulate constitutive testing strain control pure shear use material # 1
confinement_strain = 0.0
strain_increment_size = incr_size
maximum_strain = max_strain
number_of_increment = num_of_increm;
bye;
```

Computed G/Gmax curve exactly matches the one used for input at control points.

The difference in G/Gmax between control points can be reduced by using more than just 9 control points as in this example.
Figure 707.65: Stress-Strain Relationship

Figure 707.66: The $G/G_{\text{max}}$ results.
Figure 707.67: Damping Ratio Plot
707.25 Multi-yield-surface Drucker-Prager for G/Gmax plot

Problem description:

This model illustrates the G/Gmax input to multi-yield-surface Drucker-Prager material. Purely deviatoric plastic flow is used in this material, which means that the parameter dilation_scale is set to zero. If user wants to model change of volume (dilation or compression) for this material, then G/Gmax curve need to be iterated upon manually by changing yield surface size directly, which is done using different DruckerPragerMultipleYieldSurface command. This example is based on one Gauss-point which use multi-yield-surface Drucker-Prager material. The G/Gmax is converted to the yield-surface size and hardening parameter inside the DSL.

ESSI model fei/DSL file:

```plaintext
model name "G/Gmax";
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 = 
    "1,0.995,0.966,0.873,0.787,0.620,0.109,0.063"
  ShearStrainGamma = 
    "0,1E-6,1E-5,5E-5,1E-4, 0.0005, 0.001, 0.005, 0.01"
;
incr_size = 0.000001 ;
max_strain= 0.005 ;
um_of_increm = max_strain/incr_size -1 ;
simulate constitutive testing strain control pure shear use material # 1
  confinement_strain = 0.0
  strain_increment_size = incr_size
  maximum_strain = max_strain
  number_of_increment = num_of_increm;
bye;
```

Inside the DSL, the yield surface radius is calculated as $\sqrt{3}\sigma_y$, where $\sigma_y$ is the yield stress of the corresponding yield surface. Then, the radius is divided by the confinement to obtain the slope (opening angle).
The hardening parameter is calculated as

\[
\frac{1}{H_i'} = \frac{1}{H_i} - \frac{1}{2G}
\]  

(707.1)

where \( H_i' \) is the current hardening parameter corresponding to yield surface \( i \). \( H_i \) is the current tangent shear modulus to surface \( i \), namely, \( H_i = 2(\frac{\gamma_i+1-\gamma_i}{\gamma_i+1-\gamma_i}) \). And \( G \) is the initial shear modulus.
Figure 707.69: Nested-Yield-Surface Drucker-Prager $G/G_{\text{max}}$ results

Figure 707.70: Damping Ratio Plot
Appendix 708

Brief History of the Real-ESSI Simulator Development (1986-)
This section briefly describes history of the development of the Finite Element Interpreted, FEI, that is currently represented by the Real-ESSI Simulator system. Developments are presented chronologically, with very brief description of capabilities, and with references to further reading and documents with more information.

1986-1988: Development of the FRAME_and_GRID program, using BASIC programming language, on SHARP 1500, CASIO 1000 (48KB RAM) and ZX Spectrum (128KB RAM), by Boris Jeremić, undergraduate student at the University of Belgrade.

1988-1989: Development of the Earthquake Soil Structure Interaction (ESSI) Program in time domain for axisymmetric solids with general 3D loads, using higher modes of response in circumferential direction, expanded in Fourier series, so that any general 3D loading and deformation can be modeled, using FORTRAN programming language, on PC-DOS, x286+287, 640KB+384KB RAM, by Boris Jeremić, undergraduate student at the University of Belgrade, as part of his Diploma Thesis (Jeremić, 1989).

1989-1992: Development of the Finite Element Interpreter (FEI), a general purpose static and dynamic, elastic and elastic-plastic finite element program for solids, rudimentary parser for a simple Domain Specific Language (DSL), using C Programming language, on PC-DOS, x286+287, 640KB+384KB RAM, by Boris Jeremić, a staff engineer at (a) Energoprojekt-Hidroinžinjering Company in Belgrade, Yugoslavia, at (b) Bekhme Dam Project site in Iraq, and at (c) Gasser&Scepan Design Bureau in Baar, Switzerland.

1992-1997: Development of the program FEM, featuring small and large deformation elasto-plasticity, solids (bricks with 8, 20 and 27 nodes), solution advancement control, using C++ Programming language, on Sun-SparcStation 5, Solaris, 256MB RAM, and on PC-DOS x386, x486 and on PC-Linux-TurboRedHat, by Boris Jeremić, a graduate student at the University of Colorado at Boulder, as part of his Master Thesis (Jeremić, 1994) and PhD Dissertation (Jeremić, 1997).
1997-2000: Continued development of the program FEM, addition of dynamics from ESSI, structural elements from FRAME_and_GRID, Parallel version, MPI based, linking with FEI, using C++ Programming language, on PC-Linux, and PC-Linux cluster: NorthCountry, 4 nodes + master, 100based T network, by Professor Boris Jeremić, at Clarkson University and at the University of California at Davis.

2000-2006: Developments continued with introduction of all the previous and new developments from FEM into G3 Framework, later renamed OpenSEES, at PEER, using C++ Programming language, on PC-Linux, by Professor Boris Jeremić and co-workers at the University of California at Davis, CA, USA, see Final Report Presentation.

2006-Present: Development of the Real ESSI Simulator System (aka NRC-ESSI, MS-ESSI), using C++, FORTRAN, FEI-DSL, Python Programming languages, on PC-Linux, by Professor Boris Jeremić and co-workers at UCD. For details see main Real-ESSI Simulator web site or real-essi.us or real-essi.info (they all point to the same URL),
Appendix 709

Work Organization (1989)
This section describes in some detail work organization related to the development of FEI modeling
and computational system.

709.1 Communication

Tablets, smart phones, laptops and computers, using https://zoom.us/ as it works on linux and all
other OSs.

709.2 Writing (Notes, Code, &c.) Version Control

709.2.1 Source Code

Memory Leaks Memory leaks are best discovered by running Valgrind (http://valgrind.org/).
There are a number of tools that can be used with Valgrind. Mentioned are some of the most important
ones, with example commands\(^1\)

use of tcsh is assumed, with a time stamp (used in commands below) set as: set_TIMESTAMP ←
= `date +%h_%d_%Y_%Hh_%Mm_%Ss__%A`

- (time valgrind --tool=cachegrind $argv[1] >! $argv[1].cachegrind.$TIMESTAMP.out)>>!
  $argv[1].cachegrind.$TIMESTAMP.err
- (time valgrind --tool=callgrind $argv[1] >! $argv[1].callgrind.$TIMESTAMP.out)>>!
  $argv[1].callgrind.$TIMESTAMP.err
- (time valgrind --tool=massif $argv[1] >! $argv[1].massif.$TIMESTAMP.out)>>!
  $argv[1].massif.$TIMESTAMP.err
- (time valgrind --tool=memcheck --leak-check=full --show-reachable=yes --freelist-vol=1000000
- valgrind -v --leak-check=yes --show-reachable=yes --num-callers=32 --trace-malloc=yes ←
  --error-limit=no --tool=massif $argv[1]

\(^1\)Examples use synthax from few years ago, so should be proper synthax should be verified using excellent Valgrind
documentation.
709.2.2 Lecture Notes

Maintain lecture notes using git on https://github.com/.

Checking all http links in lecture notes using script ESSI_check_URLs_in_lecture_notes.sh in bin.

709.2.3 Bibliography

Bibliography List.

Papers of interest are organized in bibtex files (managed through git version control).

A list of those paper is compiled and available at:


Bibliography Repository.

Most listed papers are available at:
http://sokocalo.engr.ucdavis.edu/~jeremic/PAPERSlocalREPO/. This site is only accessible to members of the Computational Mechanics group at University of California at Davis, and few other collaborating entities.
709.3 Backup

709.4 Calendar

709.5 Useful Programs and Scripts

709.5.1 Backup Scripts

709.5.2 Domain Reduction Method Processing Programs and Scripts

DRM Node Extraction for fk.

fk Output Processing for DRM.

709.5.3 Pre Processing Programs and Scripts

709.5.4 Post Processing Programs and Scripts

709.5.5 Parallel Computer Architecture

http://www.open-mpi.org/projects/hwloc/
Appendix 710

Collected Bibliography

Compilation of all collected bibliography, over years, not necessarily cited in this book.
Bibliography

by:

Jeremić CompMech Group
Department of Civil and Environmental Engineering
University of California, Davis
Bibliography


[Ala] Fadel Alameddine. Private communications. ..., 2003 –.


[Ala ] Fadel Alameddine. Private communications. ..., 2003 –.


Jeremić et al., UCD and LBNL version: 3. August, 2020, 19:54


A.TXX: L. A.Taber. Application of shell theory to cardiac mechanics. XX, XX.


BA08: D. M. Boore and G. M. Atkinson. Ground-motion prediction equations for the average horizontal component of pga, pgv, and 5 %-damped psa at spectral periods between 0.01 s and 10.0 s. *Earthquake Spectra*, March 2008.


[CB08] Kenneth W Campbell and Yousef Bozorgnia. NGA ground motion model for the geometric mean horizontal component of PGA, PGV, PGD and 5% damped linear elastic response spectra for periods ranging from 0.01 to 10 s. Earthquake Spectra, 24(1):139–171, 2008.


[FD05] Gregory Fenves and Mathew Dryden. Nees sfsi demonstration project. NEES project meeting, TX, Austin, August 2005.


