# Contents

# 1 Elastic Beam Element for a Simple Frame Structure

**Problem description:**

- Structure size

  Structure Width=6m, Height=6m, Force=100N

- Element size

  Element length=6m, width=1m, height=1m, $\rho = 0.0$, E=1E8Pa, $\nu = 0.0$.

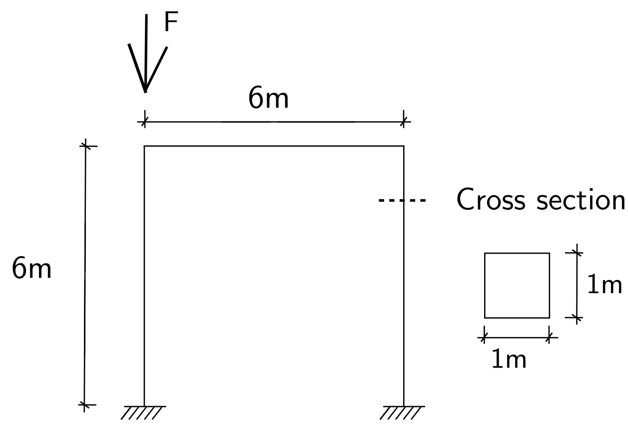


Figure 1: Problem description for the presentation example with *beam_ elastic* element

**ESSI model fei file:**

```
model name "beam_element_presentation" ;

add node #    1 at (        0.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    2 at (        0.000000*m,        0.000000*m,        6.000000*m) with 6 dofs;
add node #    3 at (        6.000000*m,        0.000000*m,        6.000000*m) with 6 dofs;
add node #    4 at (        6.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;

elastic_constant = 1e8*N/m^2;
b=1*m;
h=1*m;
rho   = 0*kg/m^3;    // Mass density
add element # 1 type beam_elastic with nodes (1, 2)
   cross_section = b*h
   elastic_modulus = elastic_constant
   shear_modulus = elastic_constant/2
   torsion_Jx = 0.33*b*h^3
   bending_Iy = b*h^3/12
   bending_Iz = h*b^3/12
   mass_density = rho
   xz_plane_vector = (1, 0, 1 )
   joint_1_offset = (0*m, 0*m, 0*m )
```

```
   joint_2_offset = (0*m, 0*m, 0*m );
add element # 2 type beam_elastic with nodes (2,3)
   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 files for this example can be downloaded here.

## 2 ShearBeam Element for Pisano Materials

**Problem description:**

In the element type "ShearBeamLT", only one Gauss point exists. So the one ShearBeamLT element was used here to test the Pisano[1] materials.

As shown in the figure (2), only one Gauss point exists in the middle of ShearBeam element. The vertical force $F_z$ was used as the confinement for the Gauss point. And the back and forth force $F_x$ was used as loading and unloading for the Gauss point.

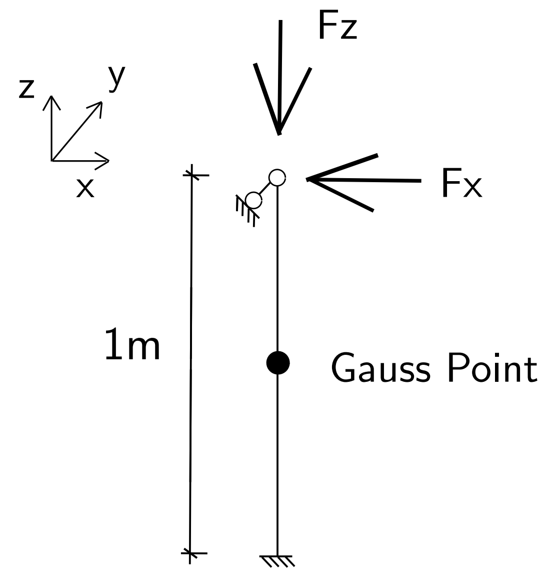


Figure 2: ShearBeam element to test the plastic materials

[1] Federico Pisanò and Boris Jeremić. Simulating stiffness degradation and damping in soils via a simple visco-elastic–plastic model. Soil Dynamics and Earthquake Engineering, 63:98–109, 2014

**Results**

The stress-strain relationship was plotted in Fig.(3).

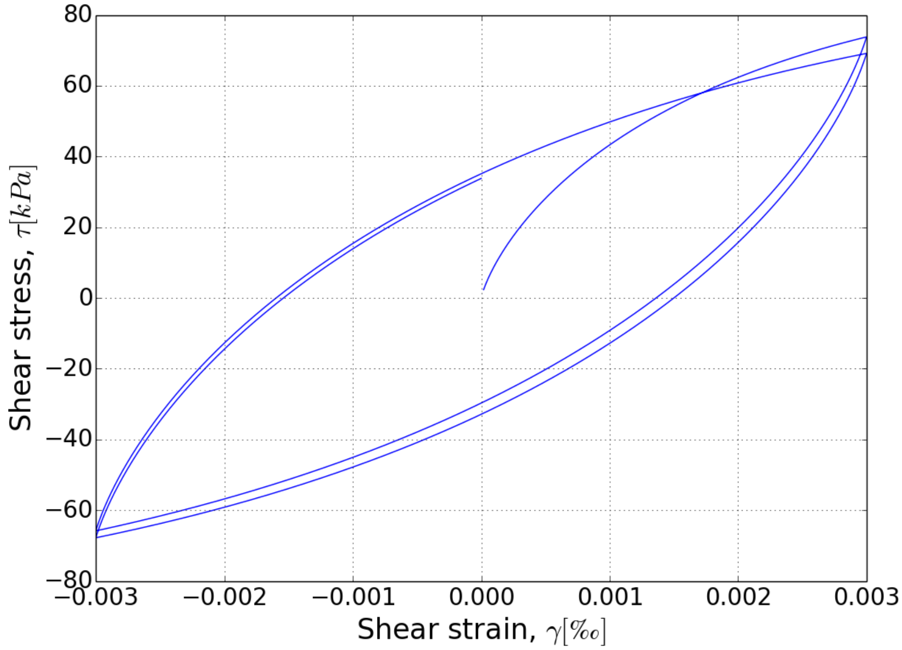


Figure 3: Shear stress-strain relationship in the Pisano model

**ESSI model fei file:**

```
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;
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;
define algorithm With_no_convergence_check ;
define solver UMFPack;
simulate 2000 steps using static algorithm;

bye;
```

The ESSI model fei files for this example can be downloaded here.

# 3 4NodeANDES Cantilever Beams Under the Force Perpendicular to Plane

**Problem description:**

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

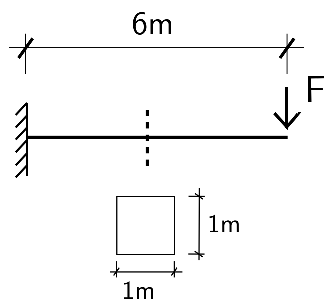


Figure 4: Problem description for cantilever beams

**Numerical model:**

When the force direction is perpendicular to the plane, only the bending deformation is calculated in 4NodeANDES elements.

The 4NodeANDES elements were shown in Figure (5).

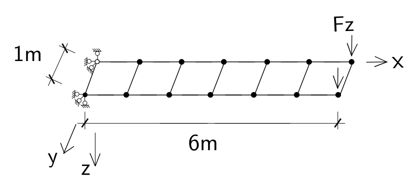


Figure 5: 4NodeANDES elements for cantilever beams under force perpendicular to plane

**ESSI model fei file:**

```
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 (        0.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    2 at (        6.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    3 at (        1.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    4 at (        2.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    5 at (        3.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    6 at (        4.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
```

```
add node #    7 at (        5.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    8 at (        6.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #    9 at (        0.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   10 at (        5.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   11 at (        4.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   12 at (        3.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   13 at (        2.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   14 at (        1.000000*m,        1.000000*m,        0.000000*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 files for this example can be downloaded here.

# 4 4NodeANDES Cantilever Beams under the Inplane Force

**Problem description:**

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

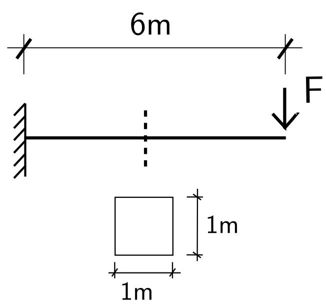


Figure 6: Problem description for cantilever beams

**Numerical model:**

When the force direction is inplane, both the bending and shear deformation are calculated in 4Node-ANDES elements.

The 4NodeANDES elements under inplane force were shown in Figure (7).

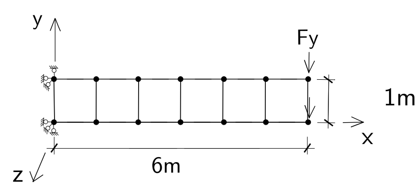


Figure 7: 4NodeANDES elements for cantilever beams under inplane force

**ESSI model fei file:**

```
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 (        0.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #     2 at (        6.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #     3 at (        1.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #     4 at (        2.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #     5 at (        3.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #     6 at (        4.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #     7 at (        5.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #     8 at (        6.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
```

```
add node #    9 at (        0.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   10 at (        5.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   11 at (        4.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   12 at (        3.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   13 at (        2.000000*m,        1.000000*m,        0.000000*m) with 6 dofs;
add node #   14 at (        1.000000*m,        1.000000*m,        0.000000*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 files for this example can be downloaded here.

# 5 27NodeBrick Cantilever Beam for the static load

**Problem description:**

Length=6m, Width=1m, Height=1m, Force=100N, E=1E8Pa, $\nu = 0.0 - 0.49$. The force direction was shown in Figure (8).

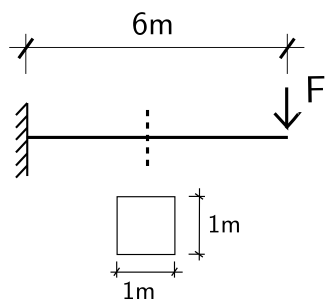


Figure 8: Problem description for cantilever beams

**Numerical model:**

The 27NodeBrick elements for cantilever beams were shown in Figure (9):

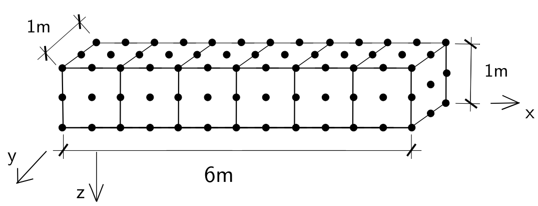


Figure 9: 27NodeBrick elements for cantilever beams

**ESSI model fei file:**

```
model name "6meter_cantilever_27brick" ;

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

add node #      1 at (  0.0000 *m,  1.0000 *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 (  6.0000 *m,  1.0000 *m, 0.0000 *m) with 3 dofs;
add node #      4 at (  5.0000 *m,  1.0000 *m, 0.0000 *m) with 3 dofs;
add node #      5 at (  4.0000 *m,  1.0000 *m, 0.0000 *m) with 3 dofs;
add node #      6 at (  3.0000 *m,  1.0000 *m, 0.0000 *m) with 3 dofs;
...
...
add node #    117 at (  5.5000 *m,  0.5000 *m, 1.0000 *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,     30,     51,     52,     53,     34,     38,     54,     55,     39,     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 files for this example can be downloaded here.

# 6 27NodeBrick Cantilever Beams for 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 was a dynamic input.

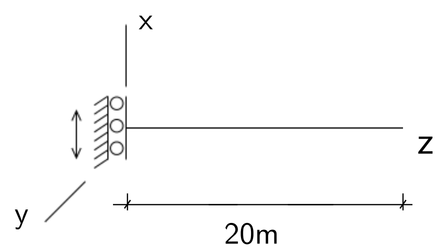


Figure 10: Problem description for one simple dynamic example

**Numerical model:**

The numerical model applied 27NodeBrick to simulate the 1D motion.

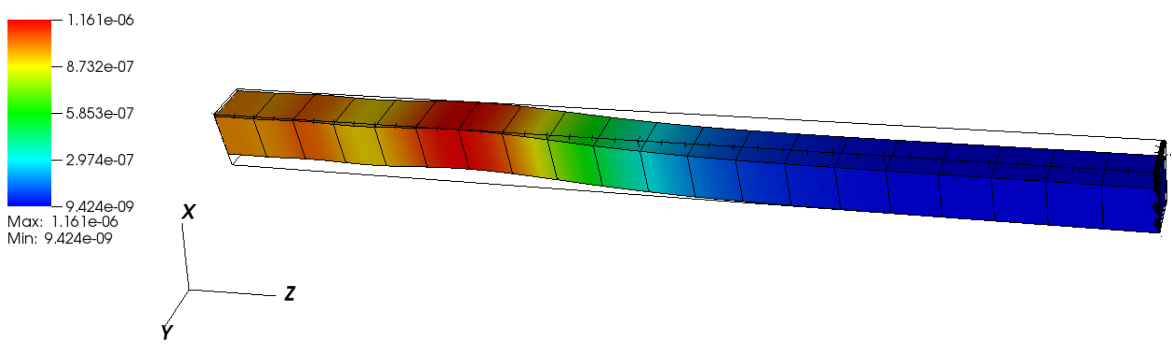


Figure 11: Numerical model for one simple dynamic example

**ESSI model fei file:**

```
model name "dynamic_example";

add material # 1 type linear_elastic_isotropic_3d_LT
   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 (27,21,19,25,9,3,1,7,24,20,22,26,6,2,4,8,18,12,10,16,14,15,11,13,17,23,5) 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 (81,75,73,79,63,57,55,61,78,74,76,80,60,56,58,62,72,66,64,70,68,69,65,67,71,77,59) use material # 1 ;
add element # 5 type 27NodeBrickLT with nodes (99,93,91,97,81,75,73,79,96,92,94,98,78,74,76,80,90,84,82,88,86,87,83,85,89,95,77) use material # 1 ;
...
...
add element # 20 type 27NodeBrickLT with nodes (369,363,361,367,351,345,343,349,366,362,364,368,348,
  344,346,350,360,354,352,358,356,357,353,355,359,365,347) 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 # 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 files for this example can be downloaded here.

# 7 4NodeANDES Square Plate with Four Edges Clamped

**Problem description:**

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

The four edges are **clamped**.

The load is the uniform normal pressure on the whole plate. Self weight is used to apply the uniform normal pressure.

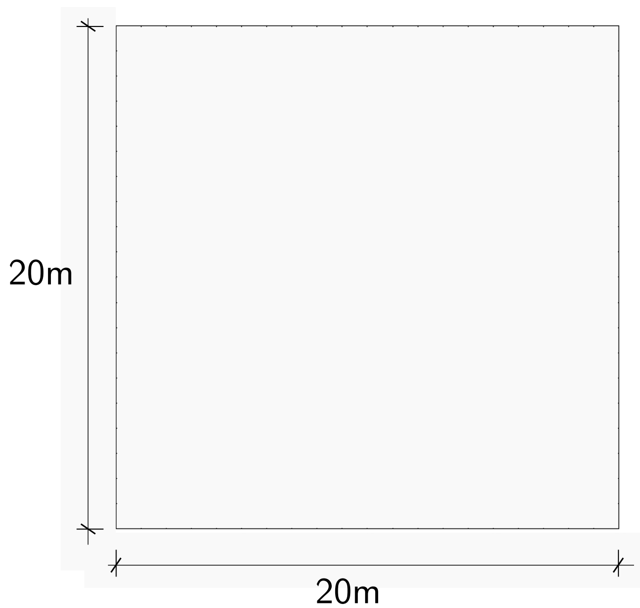


Figure 12: Square plate with four edges clamped

**Numerical model:**

The element side length is 1 meter.

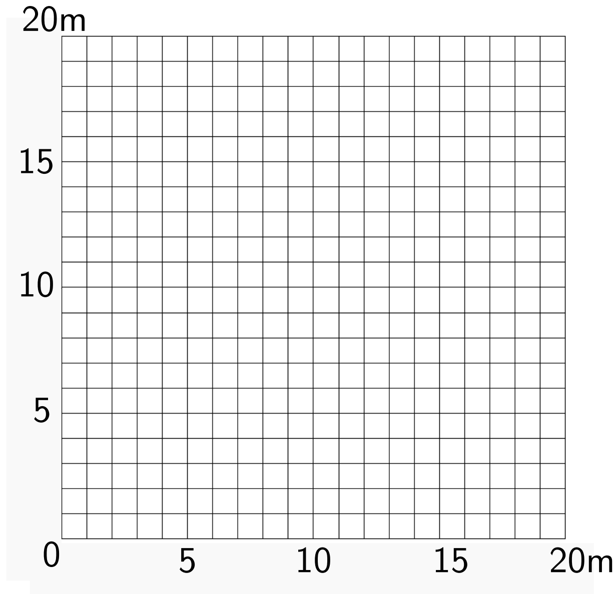


Figure 13: 4NodeANDES edge clamped square plate with element side length 1m

**ESSI model fei 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 (        0.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    2 at (       20.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    3 at (        1.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    4 at (        2.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    5 at (        3.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
add node #    6 at (        4.000000*m,        0.000000*m,        0.000000*m) with 6 dofs;
...
...
add node #  441 at (       19.000000*m,       19.000000*m,        0.000000*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   ;
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 files for this example can be downloaded here.

# 8 One Dimensional DRM Model

Domain Reduction Method (DRM) is a finite element methodology for modeling earthquake ground motion. Please look at the reference[2] for more information.

**Problem description:**

One simple 1D DRM model is shown in Fig.(14). 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 are shown in Appendix.A.

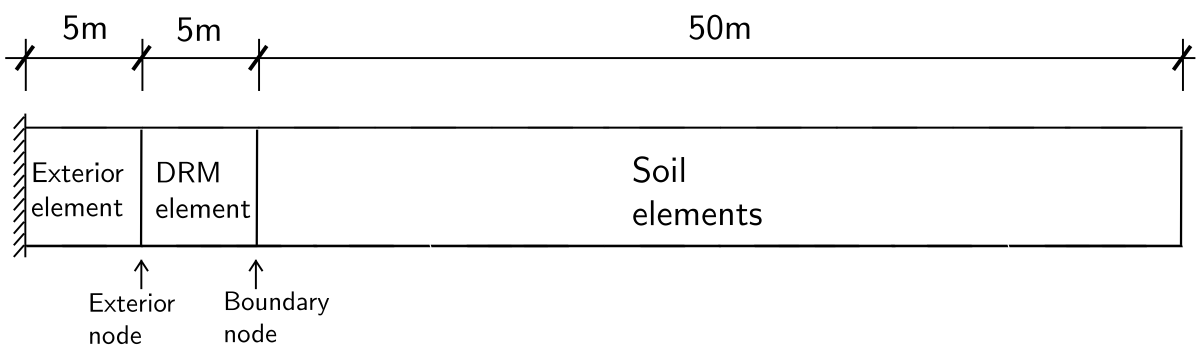


Figure 14: Program description for the 1D DRM model

**Numerical model:**

[2]Bielak, J., Loukakis, K., Hisada, Y., and Yoshimura, C. (2003). Domain reduction method for three-dimensional earthquake modeling in localized regions, Part I: Theory. Bulletin of the Seismological Society of America, 93(2), 817-824.

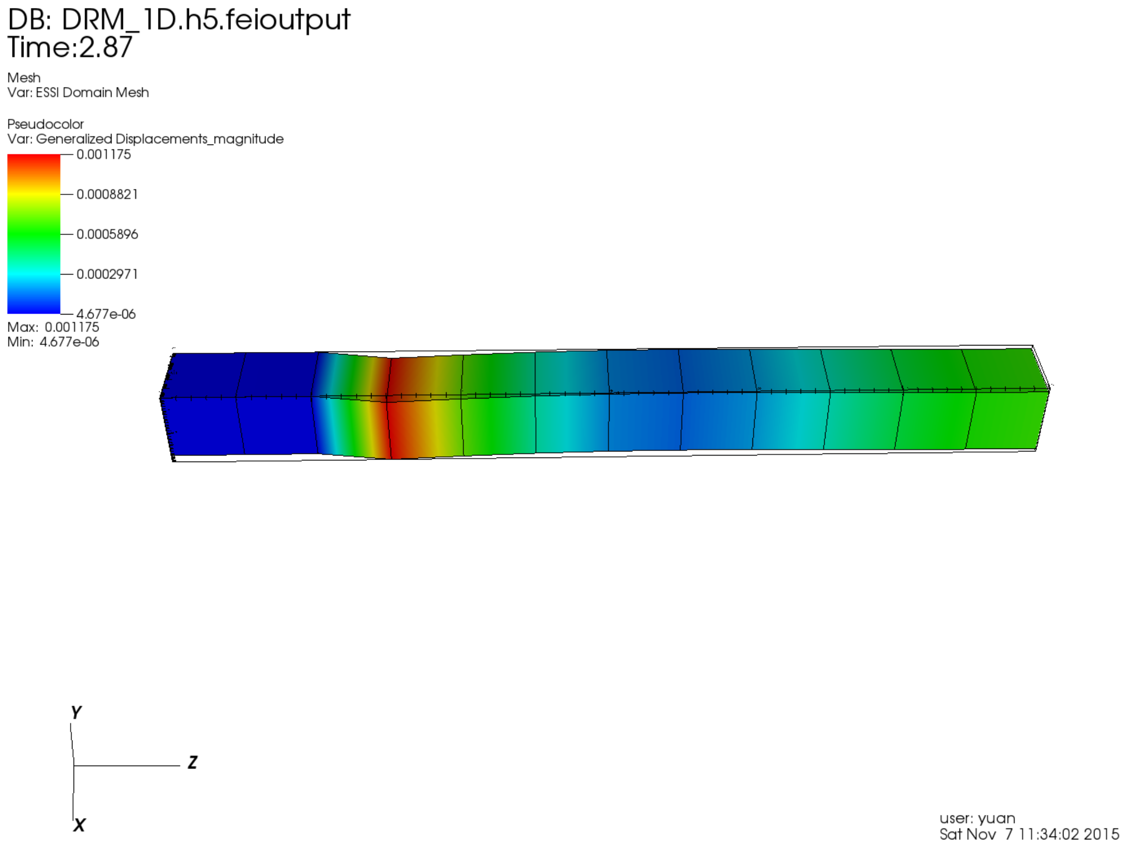


Figure 15: Diagram for the 1D DRM model

**ESSI model fei file:**

```
model name "DRM" ;

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

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

//Material for exterior layer
add material # 3 type linear_elastic_isotropic_3d_LT
  mass_density = 2000*kg/m^3
  elastic_modulus = 1300*MPa
  poisson_ratio = 0.3;
//
add node #     1 at (         0.000000*m,         0.000000*m,         0.000000*m) with 3 dofs;
add node #     2 at (         5.000000*m,         0.000000*m,         0.000000*m) with 3 dofs;
add node #     3 at (         5.000000*m,         5.000000*m,         0.000000*m) with 3 dofs;
```

```
add node #    4 at (        0.000000*m,        5.000000*m,        0.000000*m) with 3 dofs;
add node #    5 at (        5.000000*m,        0.000000*m,       50.000000*m) with 3 dofs;
add node #    6 at (        5.000000*m,        0.000000*m,        5.000000*m) with 3 dofs;
...
...
add node #   52 at (        0.000000*m,        5.000000*m,       -5.000000*m) with 3 dofs;

//
add element #     1 type 8NodeBrickLT with nodes(  1,     4,     3,     2,    24,    44,
    34,     6) use material # 1;
add element #     2 type 8NodeBrickLT with nodes( 24,    44,    34,     6,    23,    43,
    33,     7) use material # 1;
...
add element #    12 type 8NodeBrickLT with nodes( 48,    47,    45,    46,    52,    51,
    49,    50) use material # 3;

//
fix node #    1 dofs uy   ;
fix node #    1 dofs uz   ;
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 = 0.5050
   beta = 0.2525;
simulate 999 steps using transient algorithm
   time_step = 0.01*s;

bye;
```

The ESSI model fei files for this example can be downloaded here.
The same model for this example with 27NodeBrick 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.(14) except this model use far more soil elements.

The general view is shown in Fig.(16) below.

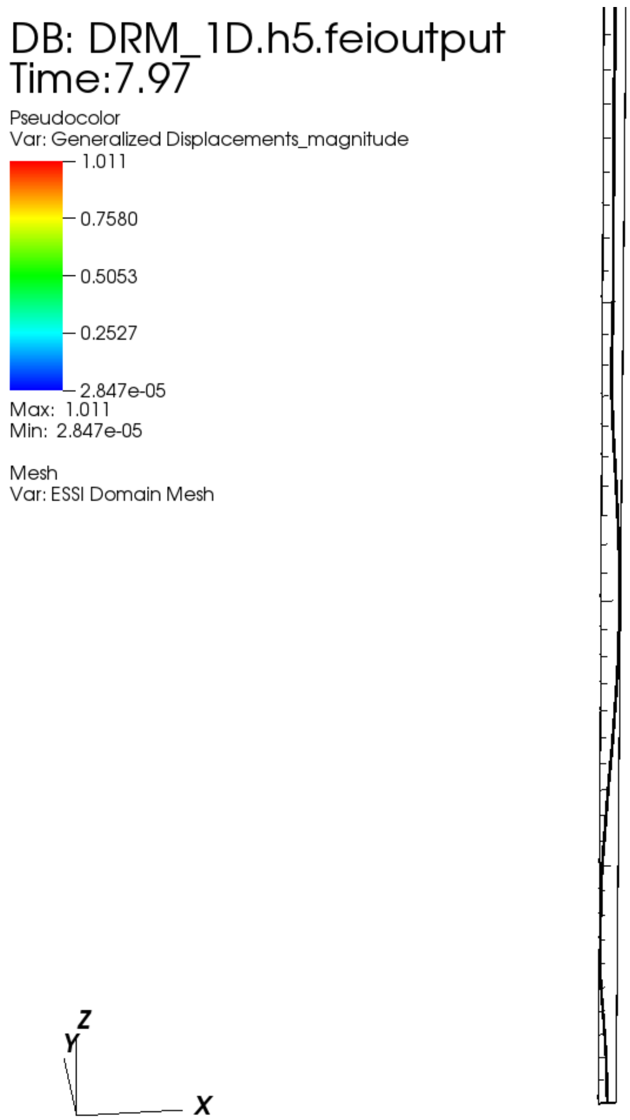


Figure 16: Long 1D DRM model

There is still now outgoing waves at the exterior layers, which is shown in Fig(17).

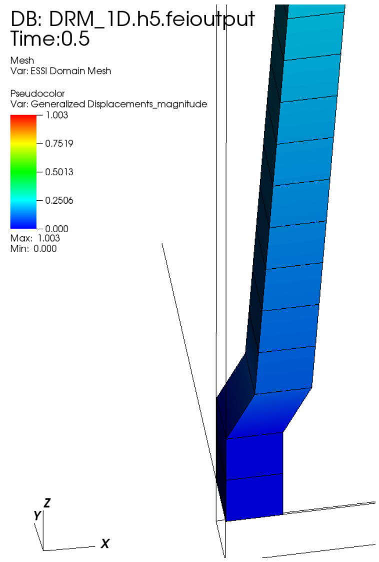


Figure 17: Long 1D DRM model: exterior layer

The ESSI model fei files for this example can be downloaded here.
The results can also be seen from this video.

# 9 Three Dimensional DRM Model

**Problem description:**

As shown in Fig.(18), the DRM layer is used to add the earthquake motion.

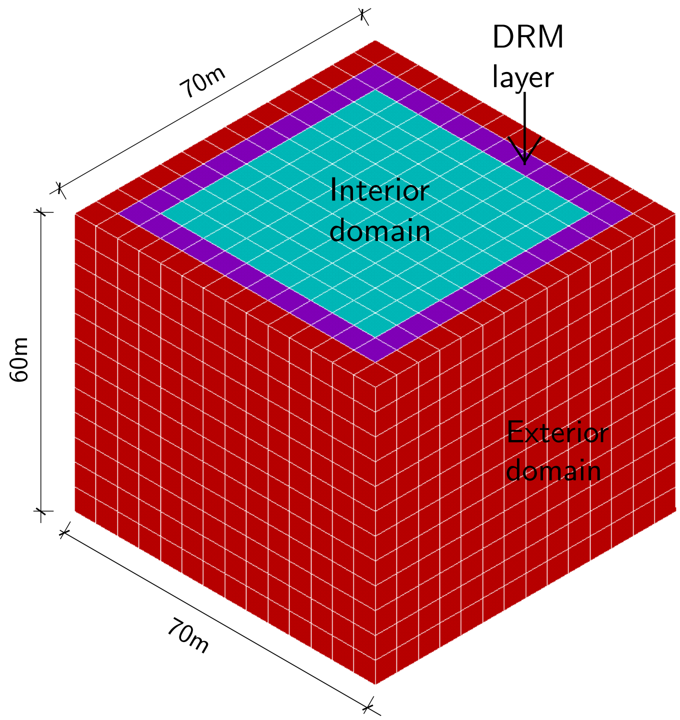


Figure 18: The diagram for 3D Domain Reduction Method (DRM)

**Numerical result:**

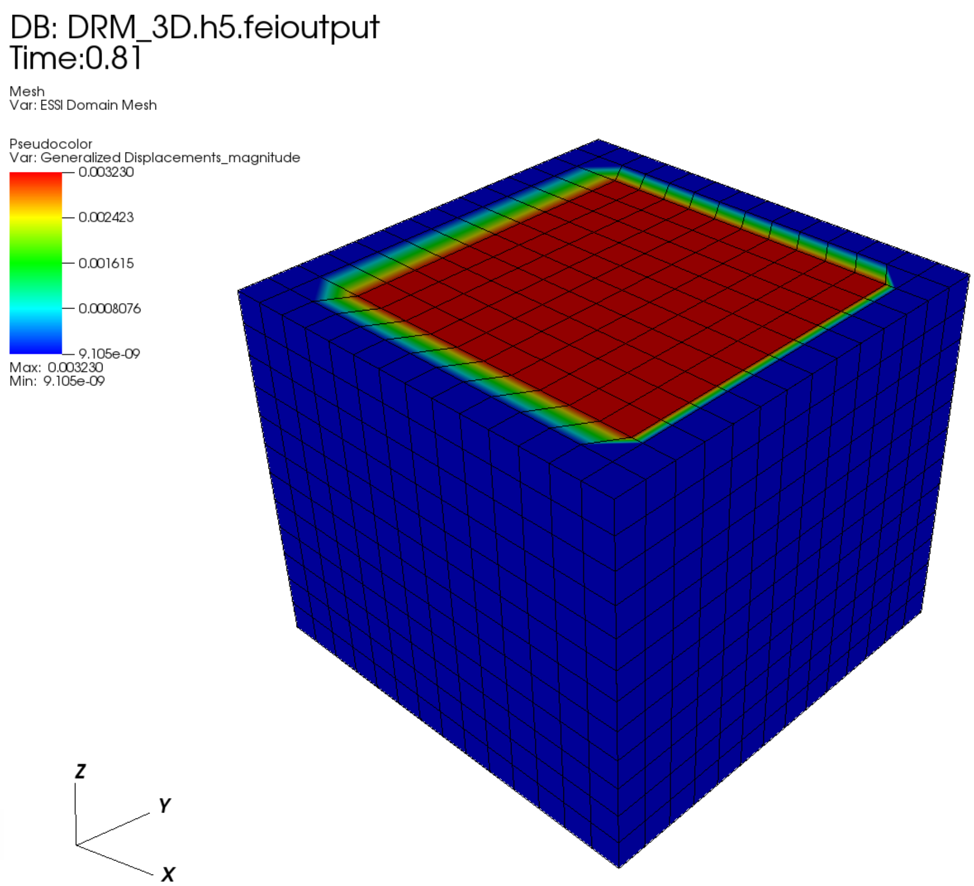


Figure 19: Diagram for the 3D DRM model

**ESSI model fei file:**

```
model name "DRM" ;

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

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

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

//
add node #     1 at (        0.000000*m,        0.000000*m,        0.000000*m) with 3 dofs;
```

```
add node #    2 at (       50.000000*m,        0.000000*m,        0.000000*m) with 3 dofs;
add node #    3 at (        5.000000*m,        0.000000*m,        0.000000*m) with 3 dofs;
add node #    4 at (       10.000000*m,        0.000000*m,        0.000000*m) with 3 dofs;
add node #    5 at (       15.000000*m,        0.000000*m,        0.000000*m) with 3 dofs;
add node #    6 at (       20.000000*m,        0.000000*m,        0.000000*m) with 3 dofs;
add node #    7 at (       25.000000*m,        0.000000*m,        0.000000*m) with 3 dofs;
...
...
add node #  2925 at (       55.000000*m,       55.000000*m,       -5.000000*m) with 3 dofs;

//
add element #     1 type 8NodeBrickLT with nodes(   1,    40,    41,     3,   150,   441,
    603,   151) 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 = 0.5050
  beta = 0.2525;

simulate 999 steps using transient algorithm
   time_step = 0.01*s;

bye;
```

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

# 10 References

## References

[1] Jacobo Bielak, Kostas Loukakis, Yoshiaki Hisada, and Chiaki Yoshimura. Domain reduction method for three-dimensional earthquake modeling in localized regions, part i: Theory. *Bulletin of the Seismological Society of America*, 93(2):817–824, 2003.

[2] E Faccioli, M Vanini, R Paolucci, and M Stupazzini. Comment on "domain reduction method for three-dimensional earthquake modeling in localized regions, part i: Theory," by j. bielak, k. loukakis, y. hisada, and c. yoshimura, and "part ii: Verification and applications," by c. yoshimura, j. bielak, y. hisada, and a. fernández. *Bulletin of the Seismological Society of America*, 95(2):763–769, 2005.

[3] Federico Pisanò and Boris Jeremić. Simulating stiffness degradation and damping in soils via a simple visco-elastic–plastic model. *Soil Dynamics and Earthquake Engineering*, 63:98–109, 2014.

[4] Chiaki Yoshimura, Jacobo Bielak, Yoshiaki Hisada, and Antonio Fernández. Domain reduction method for three-dimensional earthquake modeling in localized regions, part ii: Verification and applications. *Bulletin of the Seismological Society of America*, 93(2):825–841, 2003.

# 11 Appendix

**Appendix.A**

**How to make the DRM input in HDF5 format?**

As shown in Fig.(20), eight things are required in the DRM input.

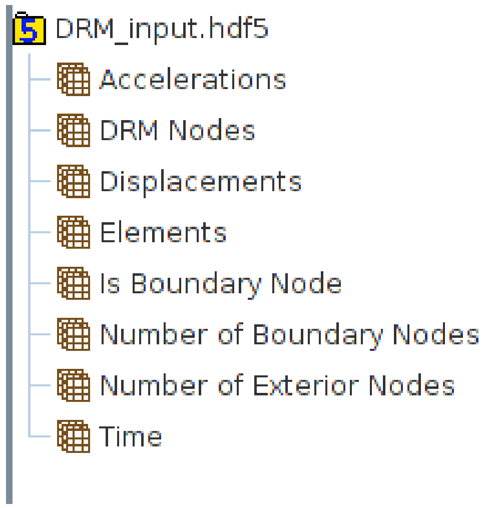


Figure 20: The HDF5 format for DRM input

The name of the subfolders must be exactly same with the name designated here.

1. Elements.

   Elements are the element number of DRM elements, which are used to add the earthquake motion.

2. DRM Nodes.

   DRM Nodes are the node number of the DRM elements.

3. Is Boundary Node.

   "Is Boundary Node" is used to describe that whether the nodes in "DRM Nodes" are the boundary nodes or the exterior nodes.

   If this value is "1", the corresponding node in "DRM Nodes" is a boundary node.

   If this value is "0", the corresponding node in "DRM Nodes" is an exterior node in the DRM element.

4. Number of Boundary Nodes.

   This is the number of boundary nodes.

5. Number of Exterior Nodes.

   This is the number of exterior nodes.

6. Displacements.

   Displacements are the displacement components of the input earthquake motion on the corresponding DRM Nodes. Displacements are a 2D array, in which the column number is the timestep number. And each row represents the displacement in one direction. If every node has 3 degrees of freedom

(DOFs) in terms of ux, uy, and uz, the first three rows represent the input displacements on the first DRM node in the directions of ux, uy, and uz. Then, the next three rows represent the input displacements for the next node. So the total row number should be three times of the number of DRM Nodes.

7. Accelerations.

   Accelerations have the same data structure with the displacements.

8. Time.

   This is the real time for each timestep in the input earthquake motion.

### Python and Matlab script to generate the DRM HDF5-based input

You can either use Python or Matlab to generate the DRM HDF5-based input.

Python script:

```
#!/bin/usr/python
# Created by Jose
# This file reads old-format DRM input files and translates them into new HDF5-based format.
#

# This file produces a rigid body input to the DRM layer. That is, all DRM nodes have same X-direction
# displacement and acceleration. In this case a sine wave is used. This is not realistic, its
# just for demonstration purposes. DRM won't work in this case but can be used to verify input if
# a pseudo-static analysis is done (zero density on all elements and apply loads with transient
# analysis.)

import scipy as sp
import h5py
import time

#Write elements and nodes data
elements = sp.loadtxt("DRMelements.txt",dtype=sp.int32)
exterior_nodes = sp.loadtxt("DRMexterior.txt",dtype=sp.int32)
boundary_nodes = sp.loadtxt("DRMbound.txt",dtype=sp.int32)

Ne = sp.array(exterior_nodes.size)
Nb = sp.array(boundary_nodes.size)

Nt = Ne+Nb

all_nodes = sp.hstack((boundary_nodes, exterior_nodes))
is_boundary_node = sp.zeros(Nt, dtype=sp.int32)
is_boundary_node[0:Nb] = 1

h5file = h5py.File("input.hdf5","w")

h5file.create_dataset("Elements", data=elements)
h5file.create_dataset("DRM Nodes", data=all_nodes)
h5file.create_dataset("Is Boundary Node", data=is_boundary_node) #This array has 1 if the node at the corresponding position in "DRM nodes" array is a boundary node and zero if not
```

```

h5file.create_dataset("Number of Exterior Nodes", data=Ne)
h5file.create_dataset("Number of Boundary Nodes", data=Nb)

#Write timestamp (time format used is that of c "asctime" Www Mmm dd hh:mm:ss yyyy example: Tue Jan 13 10:17:09 2009)
localtime = time.asctime( time.localtime(time.time()) )
h5file.create_dataset("Created",data=str(localtime))

#Generate motions

t = sp.linspace(0,10,1001)
w = 2*sp.pi/0.5
d = sp.sin(w*t)
a = -w**2*sp.sin(w*t)

#Output accelerations, displacements and time-vector

#Format is:
#
#   Accelerations has shape [3*(N_boundary_nodes + N_exterior_nodes) , Ntimesteps]
#
#
#   component A[3*n], A[3*n+1], A[3*n+2] correspond to acceleration in X, Y, and Z directions at node
#   n. The tag corresponding to node n that of the n-th component of array "DRM Nodes"

#Time vector

h5file.create_dataset("Time", data=t)

acc = h5file.create_dataset("Accelerations", (3*Nt,len(t)), dtype=sp.double,chunks=(3,50))
dis = h5file.create_dataset("Displacements", (3*Nt,len(t)), dtype=sp.double,chunks=(3,50))

for node_index in range(Nt):
   acc[3*node_index,:] = a
   acc[3*node_index+1,:] = 0*a #Zero acceleration in y and z
   acc[3*node_index+2,:] = 0*a
   dis[3*node_index,:] = d
   dis[3*node_index+1,:] = 0*d #Zero displacement in y and z
   dis[3*node_index+2,:] = 0*d



h5file.close()
```

**Matlab script:**

This is a matlab function to write DRM input in HDF5 format.

ESSI-users need to define the function arguments (e.g."time", "DRMelement") by themselves according to the actual model.

```
% Created by Chao Luo
function write_DRM_hdf5(filename,DRMu,DRMAcc,DRMNodes,DRMElements,Is_b,nb,ne,time)
h5create(filename,'/Time',[length(time)]);
h5create(filename,'/Elements',length(DRMElements),'Datatype','int32');
h5create(filename,'/DRM Nodes',length(DRMNodes),'Datatype','int32');
h5create(filename,'/Number of Exterior Nodes',length(ne),'Datatype','int32');
h5create(filename,'/Number of Boundary Nodes',length(nb),'Datatype','int32');
h5create(filename,'/Is Boundary Node',length(Is_b),'Datatype','int32');
h5create(filename,'/Displacements',size(DRMu'),'ChunkSize',[50 3]);
h5create(filename,'/Accelerations',size(DRMAcc'),'ChunkSize',[50 3]);

h5write(filename,'/Time',time');
h5write(filename,'/Elements',DRMElements');
h5write(filename,'/DRM Nodes',DRMNodes');
h5write(filename,'/Number of Exterior Nodes',ne);
h5write(filename,'/Number of Boundary Nodes',nb);
h5write(filename,'/Is Boundary Node',(Is_b+0)');
h5write(filename,'/Displacements',DRMu');
h5write(filename,'/Accelerations',DRMAcc');
```

The Python and Matlab script files can be downloaded here.