

MERCURY

Optimized Software for (Single Site) Hybrid Simulation
From Pseudo Dynamics to Real Time Hybrid Simulation

Validation/Example Manual

Ver. 1

Dae-Hung Kang
Gary Haussmann
Victor E. Saouma

Fast Hybrid Testing Laboratory

<http://fht.colorado.edu>

Department of Civil Environmental and Architectural Engineering
University of Colorado, Boulder
Boulder, Colorado 80309-0428

Contents

1	Introduction	9
2	Truss, Material Nonlinearity, Static and Dynamic	9
2.1	MATLAB	9
2.2	C++	12
3	Uniaxial Concrete Nonlinear Element, Cyclic Displacement	14
3.1	MATLAB	14
3.2	C++	16
4	Uniaxial Steel Nonlinear Element, Cyclic Displacement	17
4.1	MATLAB	17
4.2	C++	19
4.3	MATLAB	20
4.4	C++	20
5	Steel Beam-columns, Fiber Section, Static, Transient (HHT and Shing)	21
5.1	MATLAB	21
5.2	C++	24
6	Zero-length and Beam Column, Nonlinear Steel Element, load control	26
6.1	MATLAB	26
6.2	C++	27
7	Zero-length Section, and Beam Column, Fiber, Nonlinear Steel Element, load control	28
7.1	C++	29
8	Beam-column, Fiber Section, Nonlinear Material, multi-d.o.f.s displacement control, Pushover Analysis	30
8.1	MATLAB	30
8.2	C++	31
9	Reinforced Concrete Beam-Column, Fiber Section, Transient Analysis	33
9.1	MATLAB	34
9.2	C++	37
9.3	MATLAB	38
9.4	C++	39
10	Full Reinforced Concrete Frame, HHT, Shing	41
10.0.1	Frame discretization and properties	41
10.1	MATLAB	41
10.2	C++	45

List of Figures

1	Ex10: Truss, ModGMP material, load control, HHT	11
2	Cyclic wave without dimension	14
3	Examples 11- 12	15
4	Examples 16- 19	18
5	Examples 20- 22	22
6	Examples 23	26
7	Examples 24	28
8	Examples 25- 26	30
9	Beam-column elements for Ex27 and Ex28	33
10	Bar slip zero-length fiber section element (?)	34
11	Output for Ex27 and Ex28	35
12	Numerical model for real-time hybrid simulation	42
13	Reinforced concrete details, ?	43
14	Shake table test of reinforce concrete frame, ?	43
15	Section properties of Ex29	43
16	Seismic excitation for Ex29	44
17	The comparison of displacement along X-dir at each node for Mercury C++ and OpenSees	53

List of Tables

1	Features of Example Problems	10
2	Material properties of concrete for Ex27 (unit: kips, in)	34
3	Material properties of concrete for Ex28 (unit: kips, in)	34
4	Material properties of reinforcement for Ex27 and Ex28 (unit: kips, in)	35
5	Property of shear spring in zero-length element for Ex27 and Ex28 (unit: kips, in)	35
6	Concrete material properties of concrete for Ex29 (unit:kips, in)	42
7	Material properties of reinforcement for Ex29 (unit: kips, in)	43
8	Property of shear spring for Ex29 (unit: kips, in)	44
9	Properties of rigid elements (unit: kips, in)	44
10	Mass properties in first floor column nodes (unit: $kips \cdot sec^2/in$)	44
11	Mass properties in column nodes except first floor columns (unit: $kips \cdot sec^2/in$)	44
12	Mass properties in beam nodes (unit: $kips \cdot sec^2/in$)	44

1 Introduction

As with any newly developed finite element code, validation is of paramount importance under normal circumstances, and even more though when the code will be driving a physical simulation where potential damage to equipment or safety hazards may result from a “bug”.

This report will thus validate Mercury with numerous example problems through comparison with its “big brother” OpenSees, (?).

It should be noted that each of the examples presented has its own computer folder which contains the OpenSees TCL file, and both input files for Mercury (Lua for the c++ version, and the .m file for the Matlab version). Also included is an excel file containing key results for comparison. The availability of these folders will hopefully encourage potential users of Mercury to explore its capabilities, and possibly contrast it with OpenSees.

Table. 1 provides a concise summary of the example validation problems. In it, we adopt the following shortcuts:

SBC:	Stiffness-based 2D beam-column
FBC1:	Flexibility-based 2D beam-column with element iteration
FBC2:	Flexibility-based 2D beam-column without element iteration
Damage1:	Anisotropic damage 1D material model without permanent strain
Damage2:	Anisotropic damage 1D material model with permanent strain
ModKP:	Modified Kent-Park material model
ModGMP:	Modified Giuffre-Monegotto-Pinto material model
Disp(dispatch):	Displacement.

2 Truss, Material Nonlinearity, Static and Dynamic

This is a transient nonlinear analysis of a truss based on the HHT algorithm (α method), and the elements are modeled by the modified Giuffre-Monegotto-Pinto materials. Truss, and results are shown in Fig. 1.

In static analysis, incremental forces that increase by -20kN from -20kN to -200kN at node 1 along the Y direction are applied.

In the second part, the transient analysis without self-weight and nodal forces is performed with HHT integration scheme.

2.1 MATLAB

```

1 %=====
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : September 2009.
6 % File name: Ex10.m
7 %=====
8 %
9 % Description
10 % 1. Static and transient analysis
11 % 2. Load control
12 % 3. Iterative method
13 % 4. Simple 2D truss element
14 % 5. General section
15 % 6. Elastic and modified Giuffre-Monegotto-Pinto material
16 %=====
17 % Select type of analysis.
18 % AnalysisType = 1: Static analysis
19 % AnalysisType = 2: Transient analysis
20 AnalysisType = 2;
21 %=====
22 % Preface
23 Unit = {'kN', 'mm'};
24 % ndim, ndofpn
25 StrMode = {2, 2};
26 %=====
27 % Control block
28 Iteration = {'static', { {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
29                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
30                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
31                        };
32                'transient', { {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
33                            {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
34                            {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
35                            };
36                };
37 if (AnalysisType == 2)
38     Integration = {'HHT', 0, -0.1, 0.3025, 0.6, 0, 0};
39     eigens = {0.02, 0.02};
40 end
41 %=====
42 % Geometry block
43 % nodtag, x, y
44 nodcoord = {1, 0, 0;
45             2, 1500, 0;
46             3, 3000, 0;
47             4, 1500, 2000;
48             5, 3000, 2000};
49 % nodtag, x, y

```

Document Section	ID	Element	Zero Length	Formulation	Section		
2	Ex10	Truss		Stiffness	General		
3	Ex11						
	Ex12						
4	Ex16						
	Ex17						
	Ex18						
5	Ex19	Beam Column	Element Section	Flexibility, With Iteration	Fiber		
	Ex20			Flexibility, No Iteration			
	Ex21						
6	Ex22				Stiffness Based	General	
7	Ex23						
8	Ex24						
8	Ex25					Flexibility, iteration	Fiber
	Ex26					Flexibility (iteration) & Stiffness	
??	Ex27				Element &		
10	Ex29						

Document Section	ID	Constitutive Model	Hardening	Static	Transient
2	Ex10	Elastic and ModGMP	Kinematic	Load control	HHT
3	Ex11	Damage	No	Cyclic disp control	HHT Shing
	Ex12	ModKP			
4	Ex16	Simplified Bilinear	Isotropic		
	Ex17	Simplified Bilinear	Kinematic		
	Ex18	ModGMP	Kinematic & Isotropic		
	Ex19	ModGMP	Isotropic		
5	Ex20	Classical Hardening	Kinematic		
	Ex21		Kinematic & Isotropic		
	Ex22				
6	Ex23	Elastic and Simplified Bilinear	No		
7	Ex24	Elastic and Bilinear	Isotropic	Pushover	
8	Ex25	Classical Hardening			
	Ex26				
9	Ex27 and Ex28	ModGMP and ModKP/Damage	Kinematic & Isotropic		HHT/Shing
10	Ex29	Multiple		Cyclic disp control	

SBC: Stiffness-based 2D beam-column
 FBC1: Flexibility-based 2D beam-column with element iteration
 FBC2: Flexibility-based 2D beam-column without element iteration
 Damage1: Anisotropic damage 1D material model without permanent strain
 Damage2: Anisotropic damage 1D material model with permanent strain
 ModKP: Modified Kent-Park material model
 ModGMP: Modified Giuffre-Monegotto-Pinto material model
 Disp(dis): Displacement.

Table 1: Features of Example Problems

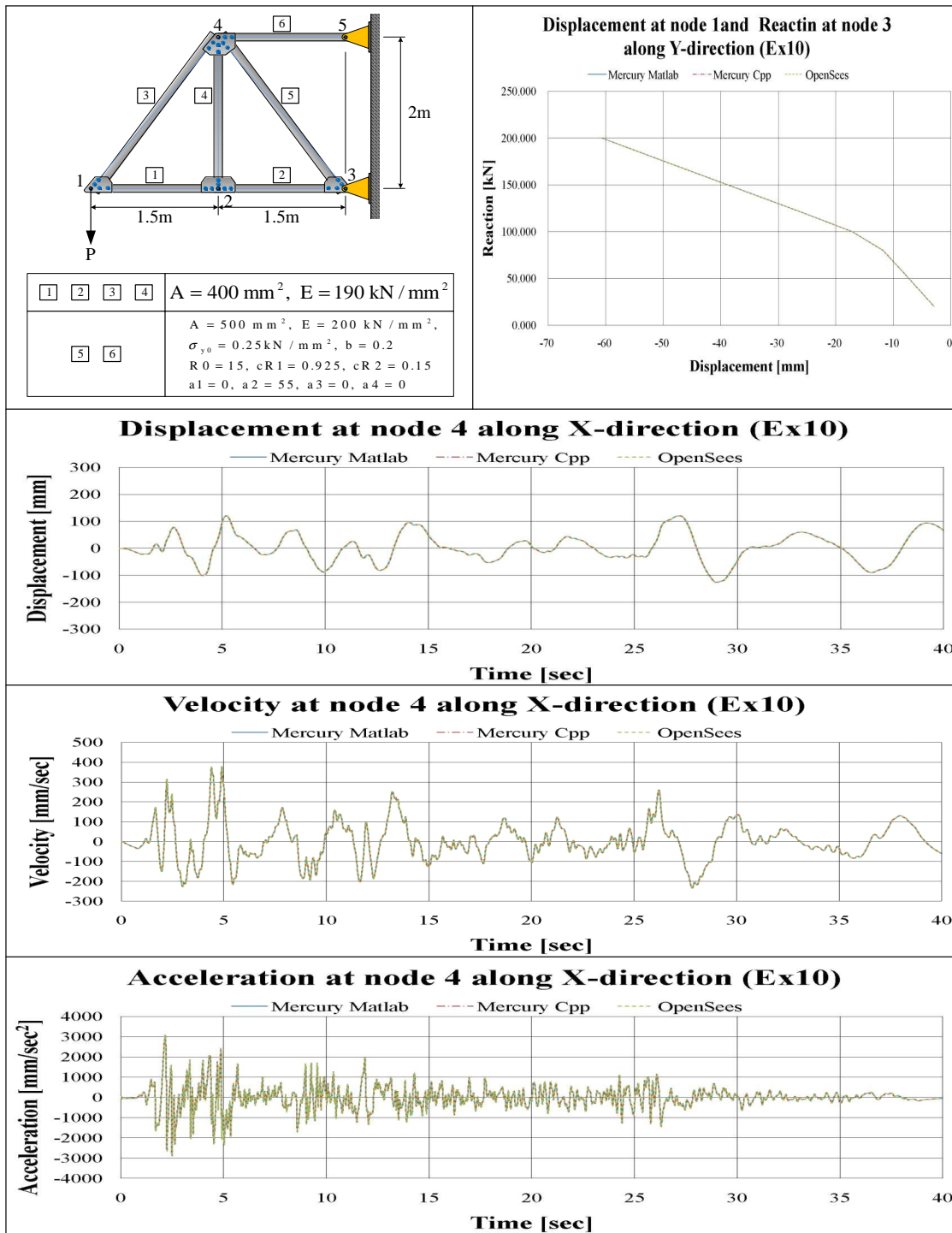


Figure 1: Ex10: Truss, ModGMP material, load control, HHT

```

50 constraint = {3, 1, 1;
51             5, 1, 1};
52 -----
53 % Element block
54 % {eletag, 'Simple2DTruss', in, jn, sectag}
55 elements = { {1, 'Simple2DTruss', 1, 2, 1};
56             {2, 'Simple2DTruss', 2, 3, 1};
57             {3, 'Simple2DTruss', 1, 4, 1};
58             {4, 'Simple2DTruss', 2, 4, 1};
59             {5, 'Simple2DTruss', 3, 4, 2};
60             {6, 'Simple2DTruss', 4, 5, 2} };
61 -----
62 % Section block
63 % sectag, 'General', {mattag, A, Ix, Iy, Iz}
64 sections = { 1, 'General', {1, 400, 0, 0, 0};
65             2, 'General', {2, 500, 0, 0, 0} };
66 -----
67 % Material block
68 % mattag, 'Elastic', E, G, density
69 materials = { {1, 'Elastic', 190, 0, 7850*10^-9}
70             {2, 'ModGMP', E, sy, b, R0, cR1, cR2, density, a1, a2, a3, a4
71             {2, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 7850*10^-9, 0, 55, 0, 55} };
72 -----
73 % Force block
74 if (AnalysisType == 1)
75     forces = { 1, 'Static', {'NodalForces', {1, 2, -20}};
76             2, 'LoadCtrl', {1, 2, {-40, -60, -80, -100, -120, -140, -160, -180, -200}} };
77 elseif (AnalysisType == 2)
78     ga = load('ElCentro_g_0_01_Matlab.txt');
79     nga = size(ga, 1);
80     for i = 1:nga
81         groundacceleration{i,1} = ga(i,1);
82         groundacceleration{i,2} = ga(i,2);
83         groundacceleration{i,3} = ga(i,3);
84     end
85     forces = { 1, 'Static', {'NodalForces', {1, 2, 0}};
86             2, 'Acceleration', {9810, groundacceleration} };
87 end
88 -----

```

2.2 C++

```

1  --- *****
2  --- AnalysisType = 1: Static analysis
3  --- AnalysisType = 2: Transient analysis
4  AnalysisType = 2
5  --- *****
6  %nodes = { {1, 0, 0, 'mass', 6.28, 6.28},
7           {2, 1500, 0, 'mass', 7.85, 7.85},
8           {3, 3000, 0},
9           {4, 1500, 2000, 'mass', 14.915, 14.915},
10          {5, 3000, 2000} }
11 ---
12 Simple2DTruss = 'truss2d'
13 elements = { {1, Simple2DTruss, 1, 2, 400, 1},
14            {2, Simple2DTruss, 2, 3, 400, 1},
15            {3, Simple2DTruss, 1, 4, 400, 1},
16            {4, Simple2DTruss, 2, 4, 400, 1},
17            {5, Simple2DTruss, 3, 4, 500, 2},
18            {6, Simple2DTruss, 4, 5, 500, 2} }
19 ---{tag, 'modifiedGMPsteel', E, rho, fy, b, ratio, R0, cR1, cR2, a1, a2, a3, a4, sigma_init}
20 materials = { {1, 'elastic', 190, 0, 0};
21             {2, 'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 0, 55, 0, 55, 0} }
22 --- *****
23 model = StructureModel(2,2)
24 model.addNodes(nodes)
25 model.addMaterials(materials)
26 model.addElements(elements)
27 model.constrainNode(3,1,1)
28 model.constrainNode(5,1,1)
29 --- *****
30 --- Static analysis
31 if (AnalysisType == 1) then
32     print("Static analysis started\n")
33     staticloading = LoadDescription()
34     staticloading.addLoad({'incrementalnodalload', 1, 2, -20, -40, -60, -80, -100, -120, -140, -160, -180, -200})
35     --- *****
36     displ = {}
37     function displperstep(increment)
38         dx1, dy1 = model.nodeDisplacements(1)
39         dx2, dy2 = model.nodeDisplacements(2)
40         dx4, dy4 = model.nodeDisplacements(4)
41         table.insert(displ, dx1)
42         table.insert(displ, dy1)
43         table.insert(displ, dx2)
44         table.insert(displ, dy2)
45         table.insert(displ, dx4)
46         table.insert(displ, dy4)
47     end
48 ---
49     react = {}
50     function reactperstep(increment)
51         fx3, fy3 = model.nodeRestoringForces(3)
52         fx5, fy5 = model.nodeRestoringForces(5)
53         table.insert(react, fx3)
54         table.insert(react, fy3)
55         table.insert(react, fx5)
56         table.insert(react, fy5)
57     end
58 ---
59     solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
60     analysis = StaticAnalysis(solver)
61     analysis.setStructureModel(model)

```

```

62 | analysis: addcallback(displperstep,"increment")
63 | analysis: addcallback(reactperstep,"increment")
64 | analysis: solve(staticloading)
65 | -----
66 | -- Set output file
67 | function writedata6(x, fname)
68 |     local f = assert(io.open(fname,'w'))
69 |     local writenl = 0
70 |     for i,v in ipairs(x) do
71 |         f:write(v, " ")
72 |         writenl = writenl + 1
73 |         -- length of row size: writenl
74 |         if (writenl > 5) then
75 |             writenl = 0
76 |             f:write("\n")
77 |         end
78 |     end
79 |     f:close()
80 | end
81 | -----
82 | function writedata4(x, fname)
83 |     local f = assert(io.open(fname,'w'))
84 |     local writenl = 0
85 |     for i,v in ipairs(x) do
86 |         f:write(v, " ")
87 |         writenl = writenl + 1
88 |         -- length of row size: writenl
89 |         if (writenl > 3) then
90 |             writenl = 0
91 |             f:write("\n")
92 |         end
93 |     end
94 |     f:close()
95 | end
96 | -----
97 | writedata6(displ,'Ex10StaticNodalDisp-1_2-4.dat')
98 | writedata4(react,'Ex10StaticReact-3-5.dat')
99 | print("Static analysis ended\n")
100 | end
101 | -----
102 | if (AnalysisType == 2) then
103 |     print("Transient analysis started\n")
104 |     earthquakeloading = LoadDescription()
105 |     accelamp = 9810
106 |     earthquakeloading:addLoad({'groundmotion', 'ElCentro_g_0_01_OpenSees.txt', dt=0.01', 1, accelamp})
107 |     -----
108 |     displ = {}
109 |     function displvertime(time)
110 |         dx1,dy1 = model.nodeDisplacements(1)
111 |         dx2,dy2 = model.nodeDisplacements(2)
112 |         dx4,dy4 = model.nodeDisplacements(4)
113 |         table.insert(displ, dx1)
114 |         table.insert(displ, dy1)
115 |         table.insert(displ, dx2)
116 |         table.insert(displ, dy2)
117 |         table.insert(displ, dx4)
118 |         table.insert(displ, dy4)
119 |     end
120 |     -----
121 |     react = {}
122 |     function reactvertime(time)
123 |         fx3,fy3 = model.nodeRestoringForces(3)
124 |         fx5,fy5 = model.nodeRestoringForces(5)
125 |         table.insert(react, fx3)
126 |         table.insert(react, fy3)
127 |         table.insert(react, fx5)
128 |         table.insert(react, fy5)
129 |     end
130 |     -----
131 |     veloc = {}
132 |     function velocvertime(time)
133 |         vx1,vy1 = model.nodeVelocities(1)
134 |         vx2,vy2 = model.nodeVelocities(2)
135 |         vx4,vy4 = model.nodeVelocities(4)
136 |         table.insert(veloc, vx1)
137 |         table.insert(veloc, vy1)
138 |         table.insert(veloc, vx2)
139 |         table.insert(veloc, vy2)
140 |         table.insert(veloc, vx4)
141 |         table.insert(veloc, vy4)
142 |     end
143 |     -----
144 |     accel = {}
145 |     function accelvertime(time)
146 |         ax1,ay1 = model.nodeAccelerations(1)
147 |         ax2,ay2 = model.nodeAccelerations(2)
148 |         ax4,ay4 = model.nodeAccelerations(4)
149 |         table.insert(accel, ax1)
150 |         table.insert(accel, ay1)
151 |         table.insert(accel, ax2)
152 |         table.insert(accel, ay2)
153 |         table.insert(accel, ax4)
154 |         table.insert(accel, ay4)
155 |     end
156 |     -----
157 |     solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-6, iterations=10})
158 |     transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.1, 0.3025, 0.6)
159 |     transientanalysis: addcallback(displvertime, "timestep")
160 |     transientanalysis: addcallback(reactvertime, "timestep")
161 |     transientanalysis: addcallback(velocvertime, "timestep")
162 |     transientanalysis: addcallback(accelvertime, "timestep")
163 |     model:setRayleighCoefficients(0.025533,0.007826)
164 |     transientanalysis: solve(4000)
165 |     -----
166 |     function writedata6(x, fname)
167 |         local f = assert(io.open(fname,'w'))
168 |         local writenl = 0

```

```

169 for i,v in ipairs(x) do
170     f:write(v, " ")
171     writenl = writenl + 1
172     -- length of row size: writenl
173     if (writenl > 5) then
174         writenl = 0
175         f:write("\n")
176     end
177 end
178 f:close()
179 end
180 --
181 function writedata4(x, fname)
182     local f = assert(io.open(fname, 'w'))
183     local writenl = 0
184     for i,v in ipairs(x) do
185         f:write(v, " ")
186         writenl = writenl + 1
187         -- length of row size: writenl
188         if (writenl > 3) then
189             writenl = 0
190             f:write("\n")
191         end
192     end
193     f:close()
194 end
195 --
196 writedata6(displ, 'Ex10TransientNodalDisp_1_2_4.dat')
197 writedata4(react, 'Ex10TransientReact_3_5.dat')
198 writedata6(veloc, 'Ex10TransientNodalVelo_1_2_4.dat')
199 writedata6(accel, 'Ex10TransientNodalAcce_1_2_4.dat')
200 print("Transient analysis ended\n")
201 end
202 -- *****

```

3 Uniaxial Concrete Nonlinear Element, Cyclic Displacement

These examples assume concrete members and seek to compare the modified Kent-Park model with the anisotropic damage model with permanent strain when subjected to cyclic displacement statically, Fig. 2. Fig. 3 describes the static analysis with cyclic displacements applied at node 2 in the X direction.

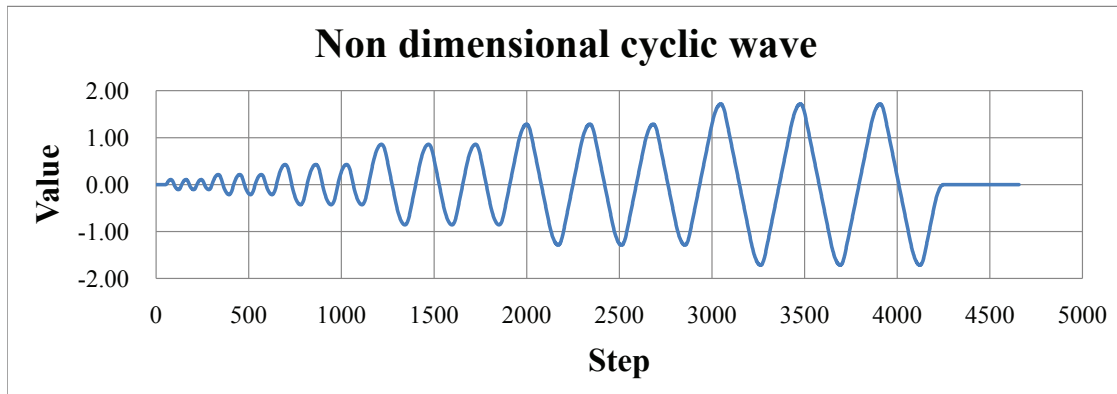


Figure 2: Cyclic wave without dimension

3.1 MATLAB

```

1 %-----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : November 2009.
6 % File name: Ex11to12.m
7 %-----
8 %
9 % Description
10 % 1. Static analysis
11 % 2. Displacement control
12 % 3. Iterative method
13 % 4. Simple 2D truss element
14 % 5. General section
15 % 6. Anisotropic damage material(Ex11)
16 % and modified Kent-Park material(Ex12)
17 %-----
18 % Select material type
19 % MatType = 1: anisotropic damage material with permanent strain (Ex11)
20 % MatType = 2: modified Kent-Park material (Ex12)
21 MatType = 1;
22 %-----

```

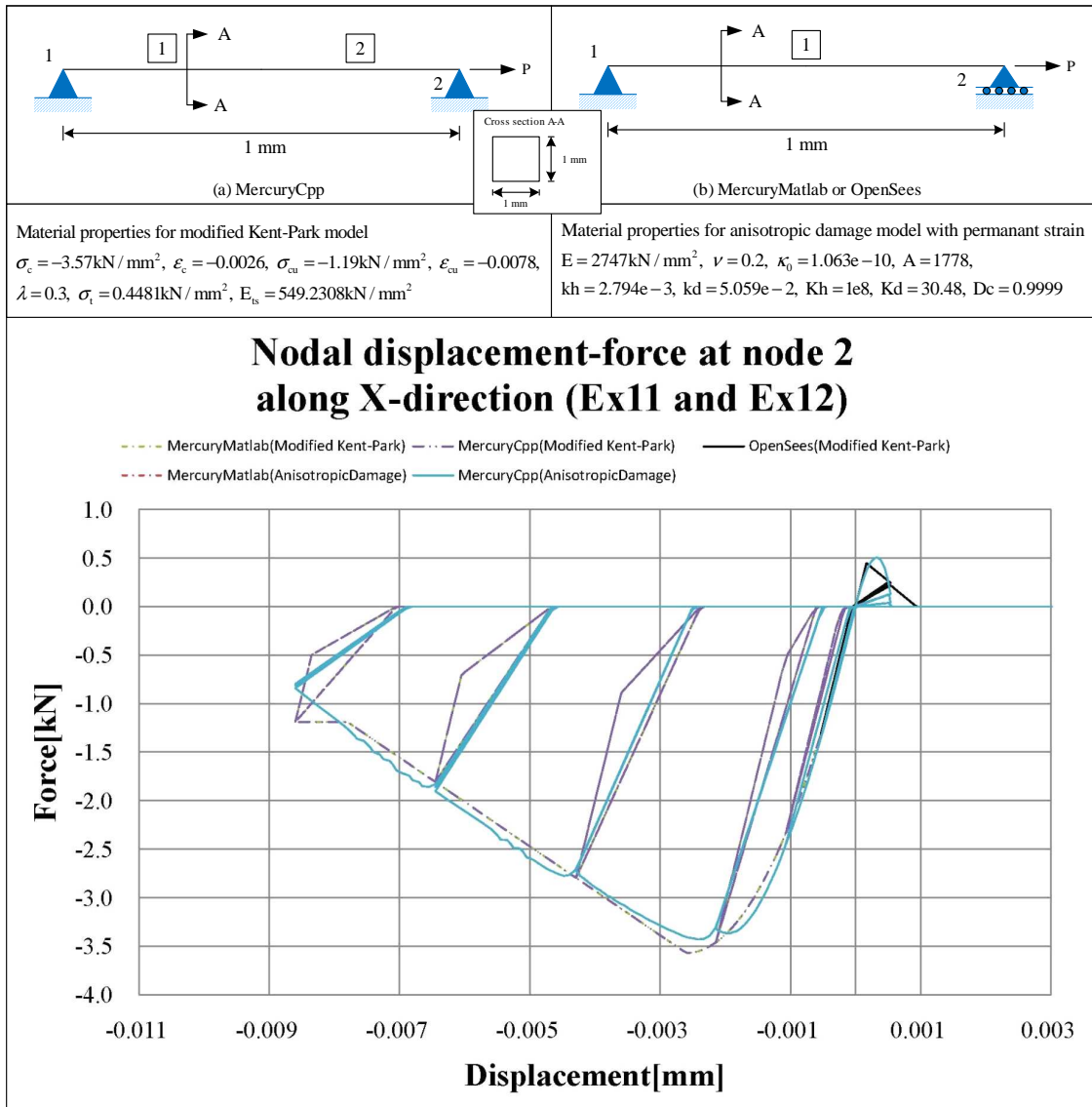


Figure 3: Examples 11- 12

```

23 % Preface
24 Unit = {'kN', 'mm'};
25 StrMode = {2, 2};
26 -----
27 % Control block
28 Iteration = {'static', { {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
29                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
30                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
31                        }
32                };
33 -----
34 % Geometry block
35 nodcoord = {1, 0, 0;
36            2, 1, 0};
37 constraint = {1, 1, 1;
38             2, 0, 1};
39 -----
40 % Element block
41 elements = {1, 'Simple2DTruss', 1, 2, 1};
42 -----
43 % Section block
44 sections = {1, 'General', {1, 1, 0, 0, 0}};
45 -----
46 % Material block
47 if (MatType == 1)
48     % {tag, 'AnisotropicDamage', E, nu, kappa0, A, kh, kd, Kh, Kd, Dc, density};
49     materials = {1, 'AnisotropicDamage', 2.747e+003, 0.2, 1.063e-010, 1.778e+003,
50                2.794e-003, 5.059e-002, 1.000e+008, 3.048e+001, 0.99999999, 0};
51 end
52 if (MatType == 2)
53     % {tag, 'ModKP', sc, ec, scu, ecu, lambda, st, Ets, density};
54     materials = {1, 'ModKP', -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077, 549.2307692, 0};
55 end
56 -----
57 % Force block
58 DispInput = load('cyclicwave.txt');
59 row = size(DispInput,1);
60 for i = 1:row
61     DispCell{i} = 0.005*DispInput(i);
62 end
63 % forcetag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
64 forces = {1, 'Static', {'NodalForces', {2, 1, 0}};
65          2, 'DispCtrl', {2, 1, DispCell}};
66 -----

```

3.2 C++

```

1  --- *****
2  --- MatType = 1: Anisotropic damage model with permanant strain (Ex11)
3  --- MatType = 2: Modified Kent-Park model (Ex12)
4  MatType = 2
5  --- *****
6  nodes = { {1, 0, 0},
7           {2, 1, 0} }
8  --- *****
9  Simple2DTruss = 'truss2d'
10 elements = {1, Simple2DTruss, 1, 2, 1, 1} }
11 --- *****
12 if (MatType == 1) then
13     materials = {{1, 'anisotropicdamage2', 2.747e+003, 0, 0.2, 1.063e-010, 1.778e+003,
14                2.794e-003, 5.059e-002, 1.000e+008, 3.048e+001, 0.99999999}};
15 end
16 if (MatType == 2) then
17     concretemat = 'ConcreteLinearTensionSoftening'
18     materials = {{1,concretemat, 549.2307692, 0, -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077}};
19 end
20 --- *****
21 model = StructureModel(2,2)
22 model:addNodes(nodes)
23 model:addMaterials(materials)
24 model:addElements(elements)
25 model:constrainNode(1,1,1)
26 model:constrainNode(2,1,1)
27 --- *****
28 function generateincrementalload()
29     --- format:          tag          node dof
30     local loadform = {'incrementalnodaldisplacement', 2, 1}
31     local f = assert(io.open('cyclicwave.txt','r'))
32     local n = f:read("number")
33     while (n ~= nil) do
34         table.insert(loadform, 0.005*n)
35         n = f:read("number")
36     end
37     f:close()
38     return loadform
39 end
40 --- *****
41 staticloading = LoadDescription()
42 --- format: 'staticnodalload' <node> <dof> <amplitude>
43 l = generateincrementalload()
44 staticloading:addLoad(l)
45 --- *****
46 --- Static analysis
47 print("Static analysis started\n")
48 displ = {}
49 function displperstep(increment)
50     dx2,dy2 = model:nodeDisplacements(2)
51     table.insert(displ, dx2)
52 end
53 ---
54 react = {}
55 function reactperstep(increment)
56     fx1,fy1 = model:nodeRestoringForces(1)

```



```

57     table.insert(react, fx1)
58 end
59 ---
60 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
61 analysis = StaticAnalysis(solver)
62 analysis:setStructureModel(model)
63 ---analysis:addcallback(displperstep,"increment")
64 ---analysis:addcallback(reactperstep,"increment")
65 analysis:solve(staticloading)
66 --- *****
67 --- Set output file
68 function writedata1(x, fname)
69     local f = assert(io.open(fname,'w'))
70     local writenl = 0
71     for i,v in ipairs(x) do
72         f:write(v, " ")
73         writenl = writenl + 1
74         --- length of row size: writenl
75         if (writenl > 0) then
76             writenl = 0
77             f:write("\n")
78         end
79     end
80     f:close()
81 end
82 ---
83 if (MatType == 1) then
84     writedata1(displ,'Ex11StaticDamageNodalDisp_2.dat')
85     writedata1(react,'Ex11StaticDamageReact_1.dat')
86 end
87 if (MatType == 2) then
88     writedata1(displ,'Ex12StaticModKPNodalDisp_2.dat')
89     writedata1(react,'Ex12StaticModKPRReact_1.dat')
90 end
91 print(" Static analysis ended\n")

```

4 Uniaxial Steel Nonlinear Element, Cyclic Displacement

Bilinear and modified Giuffre-Monegotto-Pinto constitutive models with cyclic displacement are examined next, Fig. 2. Fig. 4 shows the results of the static analysis with cyclic displacements applied at node 2 in the X direction. Ex16 has bilinear material without isotropic hardening, Ex17 has bilinear material with isotropic hardening, Ex18 has modified Giuffre-Monegotto-Pinto material without isotropic hardening, and Ex19 has modified Giuffre-Monegotto-Pinto material with isotropic hardening material.

4.1 MATLAB

```

1  %-----
2  % Mercury Matlab Version 1.0.1
3  % Written by Dae-Hung Kang, CU-NEES
4  % Copyright 2009, CU-NEES
5  % Written : October 2009.
6  % File name: Ex16to17.m
7  %-----
8  % Description
9  % 1. Static analysis
10 % 2. Displacement control
11 % 3. Iterative method
12 % 4. Simple 2D truss element
13 % 5. General sections
14 % 6. Bilinear material
15 %-----
16 % A36 Steel properties
17 % Density of 7.8 g/cm^3
18 % Minimum yield strength of 250 MPa(0.25 GPa)
19 % Ultimate tensile strength of 400-550 MPa(0.4-0.55 GPa)
20 %-----
21 % Select material type
22 % MatType = 1: Bilinear material without isotropic hardening
23 % MatType = 2: Bilinear material with isotropic hardening
24 MatType = 2;
25 %-----
26 % Preface
27 Unit = {'kN', 'mm'};
28 StrMode = {2, 2};
29 %-----
30 % Control block
31 Iteration = {'static',{ {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
32                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
33                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
34                        }};
35 %-----
36 % Geometry block
37 nodcoord = {1, 0, 0;
38            2, 1, 0};
39 constraint = {1, 1, 1;
40             2, 0, 1};
41 %-----
42 % Element block
43 elements = { {1, 'Simple2DTruss', 1, 2, 1}; };
44 %-----
45 % Section block
46 sections = { 1, 'General', {1, 1, 0, 0, 0}; };
47 %-----
48 % Material block
49 if MatType == 1
50     materials = { {1, 'Bilinear', 200, 0.25, 0.2, 0, 0, 55, 0, 55} };

```

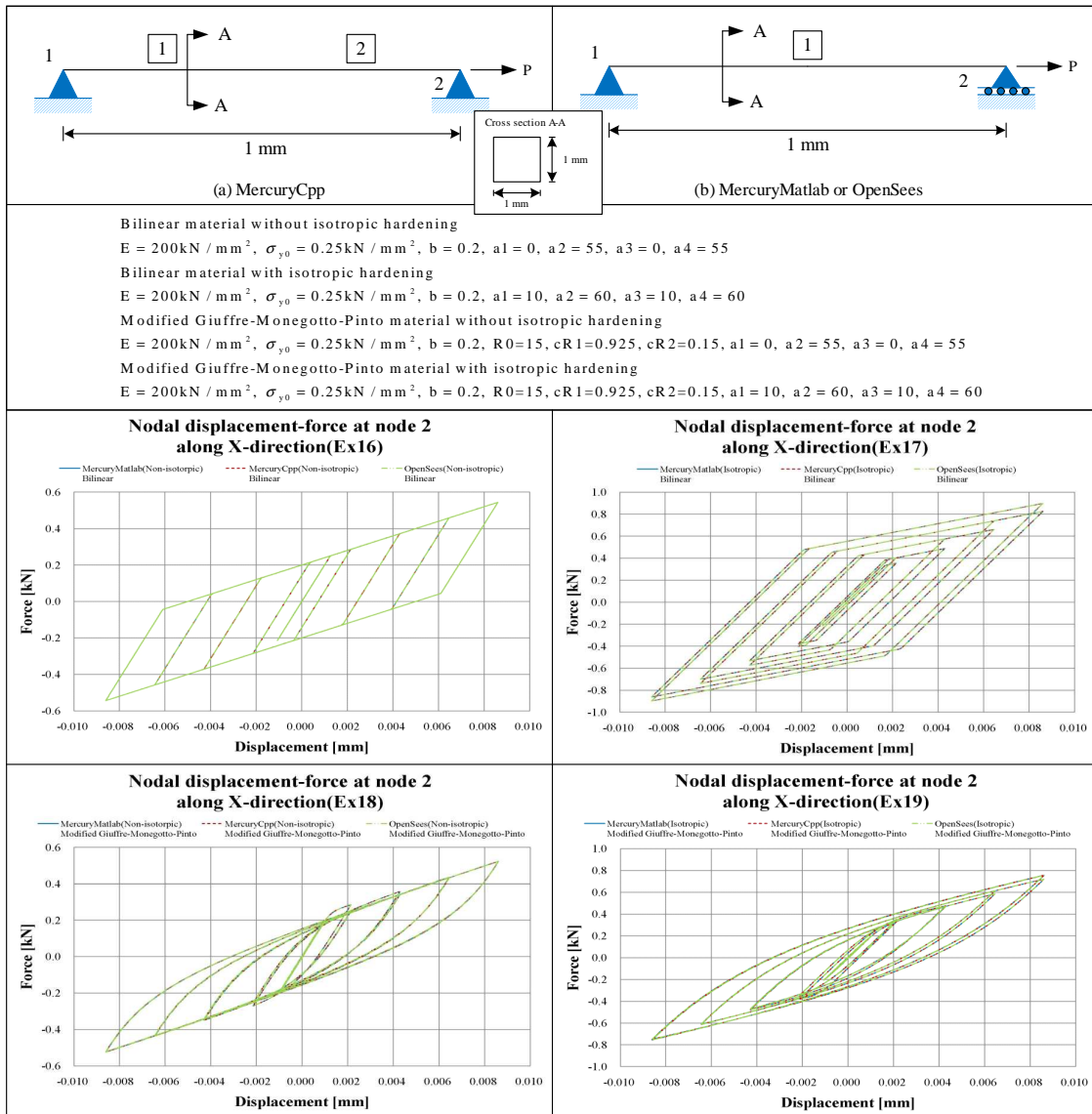


Figure 4: Examples 16- 19

```

51 elseif MatType == 2
52     materials = { {1, 'Bilinear', 200, 0.25, 0.2, 0, 10, 60, 10, 60} };
53 end
54 %-----
55 % Force block
56 DispInput = load('cyclicwave.txt');
57 row = size(DispInput,1);
58 for i = 1:row
59     DispCell{i} = 0.005*DispInput(i);
60 end
61 %         forcetag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
62 forces = { 1, 'Static', {'NodalForces', {2, 1, 0}};
63           2, 'DispCtrl', {2, 1, DispCell} };
64 %-----

```

4.2 C++

```

1  --- *****
2  --- MatType = 1: Bilinear material without isotropic hardening (Ex16)
3  --- MatType = 2: Bilinear material with isotropic hardening (Ex17)
4  MatType = 2
5  --- *****
6  nodes = { {1, 0, 0},
7            {2, 1, 0} }
8  --- *****
9  Simple2DTruss = 'truss2d'
10 elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
11 --- *****
12 if (MatType == 1) then
13     materials = {{tag, 'bilinear', E, rho, fy, b_ratio, a1, a2, a3, a4}}
14     materials = { {1, 'bilinear', 200, 0, 0.25, 0.2, 0, 55, 0, 55} }
15 end
16 if (MatType == 2) then
17     materials = { {1, 'bilinear', 200, 0, 0.25, 0.2, 10, 60, 10, 60} }
18 end
19 --- *****
20 model = StructureModel(2,2)
21 model.addNodes(nodes)
22 model.addMaterials(materials)
23 model.addElements(elements)
24 model.constrainNode(1,1,1)
25 model.constrainNode(2,1,1)
26 --- *****
27 function generateincrementalload()
28     --- format:          tag          node dof
29     local loadform = {'incrementalnodaldisplacement', 2, 1}
30     local f = assert(io.open('cyclicwave.txt','r'))
31     local n = f:read("number")
32     while (n ~= nil) do
33         table.insert(loadform, 0.005*n)
34         n = f:read("number")
35     end
36     f:close()
37     return loadform
38 end
39 --- *****
40 staticloading = LoadDescription()
41 --- format: 'staticnodalload' <node> <dof> <amplitude>
42 l = generateincrementalload()
43 staticloading:addLoad(l)
44 --- *****
45 --- Static analysis
46 print(" Static analysis started\n")
47 displ = {}
48 function displperstep(increment)
49     dx2,dy2 = model.nodeDisplacements(2)
50     table.insert(displ, dx2)
51 end
52 ---
53 react = {}
54 function reactperstep(increment)
55     fx1,fy1 = model.nodeRestoringForces(1)
56     table.insert(react, fx1)
57 end
58 ---
59 solver = NonlinearSolver(" initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
60 analysis = StaticAnalysis(solver)
61 analysis:setStructureModel(model)
62 analysis:addcallback(displperstep,"increment")
63 analysis:addcallback(reactperstep,"increment")
64 analysis:solve(staticloading)
65 --- *****
66 --- Set output file
67 function writedata1(x, fname)
68     local f = assert(io.open(fname,'w'))
69     local writenl = 0
70     for i,v in ipairs(x) do
71         f:write(v, " ")
72         writenl = writenl + 1
73         --- length of row size: writenl
74         if (writenl > 0) then
75             writenl = 0
76             f:write("\n")
77         end
78     end
79     f:close()
80 end
81 ---
82 if (MatType == 1) then
83     writedata1(displ, 'Ex16StaticNonIsotropicNodalDisp-2.dat')
84     writedata1(react, 'Ex16StaticNonIsotropicReact-1.dat')
85 end
86 if (MatType == 2) then

```

```

87 writedata1(displ,'Ex17StaticIsotropicNodalDisp_2.dat')
88 writedata1(react,'Ex17StaticIsotropicReact_1.dat')
89 end
90 print(" Static analysis ended\n")

```

4.3 MATLAB

```

1 %-----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex18to19.m
7 %-----
8 % Description
9 % 1. Static analysis
10 % 2. Displacement control
11 % 3. Iterative method
12 % 4. Simple 2D truss element
13 % 5. General sections
14 % 6. Modified Giuffre-Monegotto-Pinto material
15 %-----
16 % A36 Steel properties
17 % Density of 7.8 g/cm^3
18 % Minimum yield strength of 250 MPa(0.25 GPa)
19 % Ultimate tensile strength of 400-550 MPa(0.4-0.55 GPa)
20 %-----
21 % Select material type
22 % MatType = 1: ModGMP material without isotropic hardening (Ex18)
23 % MatType = 2: ModGMP material with isotropic hardening (Ex19)
24 MatType = 2;
25 %-----
26 % Preface
27 Unit = {'kN', 'mm'};
28 StrMode = {2, 2};
29 %-----
30 % Control block
31 Iteration = {'static',{ {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
32                       {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
33                       {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
34                       }};
35 %-----
36 % Geometry block
37 nodcoord = {1, 0, 0;
38            2, 1, 0};
39 constraint = {1, 1, 1;
40             2, 0, 1};
41 %-----
42 % Element block
43 elements = { {1, 'Simple2DTruss', 1, 2, 1}; };
44 %-----
45 % Section block
46 sections = { 1, 'General', {1, 1, 0, 0, 0}};
47 %-----
48 % Material block
49 if MatType == 1
50     mattach, 'ModGMP', E, sy, b, R0, cR1, cR2, density, a1, a2, a3, a4
51     materials = { {1, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 0, 0, 55, 0, 55} };
52 elseif MatType == 2
53     materials = { {1, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 0, 10, 60, 10, 60} };
54 end
55 %-----
56 % Force block
57 DispInput = load('cyclicwave.txt');
58 row = size(DispInput,1);
59 for i = 1:row
60     DispCell{i} = 0.005*DispInput(i);
61 end
62 %-----
63 forces = { 1, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
64          2, 'DispCtrl', {2, 1, DispCell} };
65 %-----

```

4.4 C++

```

1 --- *****
2 --- MatType = 1: ModGMP material without isotropic hardening (Ex18)
3 --- MatType = 2: ModGMP material with isotropic hardening (Ex19)
4 MatType = 2
5 --- *****
6 nodes = { {1, 0, 0},
7          {2, 1, 0} }
8 --- *****
9 Simple2DTruss = 'truss2d'
10 elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
11 --- *****
12 if (MatType == 1) then
13     materials = { {tag, 'modifiedGMPsteel', E, rho, fy, b_ratio, R0, cR1, cR2, a1, a2, a3, a4, sigma_init} }
14     materials = { {1, 'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 0.0, 55, 0, 55, 0} }
15 end
16 if (MatType == 2) then
17     materials = { {1, 'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 10, 60, 10, 60, 0} }
18 end
19 --- *****
20 model = StructureModel(2,2)
21 model:addNodes(nodes)
22 model:addMaterials(materials)
23 model:addElements(elements)

```

```

24 model:constrainNode(1,1,1)
25 model:constrainNode(2,1,1)
26 --- *****
27 function generateincrementalload()
28 --- format: tag node dof
29 local loadform = {'incrementalnodaldisplacement', 2, 1}
30 local f = assert(io.open('cyclicwave.txt','r'))
31 local n = f:read('*number')
32 while (n ~= nil) do
33 table.insert(loadform, 0.005*n)
34 n = f:read('*number')
35 end
36 f:close()
37 return loadform
38 end
39 --- *****
40 staticloading = LoadDescription()
41 --- format: 'staticnodalload' <node> <dof> <amplitude>
42 l = generateincrementalload()
43 staticloading:addLoad(l)
44 --- *****
45 --- Static analysis
46 print("Static analysis started\n")
47 displ = {}
48 function displperstep(increment)
49 dx2,dy2 = model:nodeDisplacements(2)
50 table.insert(displ, dx2)
51 end
52 ---
53 react = {}
54 function reactperstep(increment)
55 fx1,fy1 = model:nodeRestoringForces(1)
56 table.insert(react, fx1)
57 end
58 ---
59 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
60 analysis = StaticAnalysis(solver)
61 analysis:setStructureModel(model)
62 analysis:addcallback(displperstep,"increment")
63 analysis:addcallback(reactperstep,"increment")
64 analysis:solve(staticloading)
65 --- *****
66 --- Set output file
67 function writedata1(x, fname)
68 local f = assert(io.open(fname,'w'))
69 local writenl = 0
70 for i,v in ipairs(x) do
71 f:write(v, " ")
72 writenl = writenl + 1
73 --- length of row size: writenl
74 if (writenl > 0) then
75 writenl = 0
76 f:write("\n")
77 end
78 end
79 f:close()
80 end
81 ---
82 if (MatType == 1) then
83 writedata1(displ, 'Ex18StaticNonIsotropicNodalDisp_2.dat')
84 writedata1(react, 'Ex18StaticNonIsotropicReact_1.dat')
85 end
86 if (MatType == 2) then
87 writedata1(displ, 'Ex19StaticIsotropicNodalDisp_2.dat')
88 writedata1(react, 'Ex19StaticIsotropicReact_1.dat')
89 end
90 print("Static analysis ended\n")

```

5 Steel Beam-columns, Fiber Section, Static, Transient (HHT and Shing)

Beam column elements with layered section, and hardening material are analyzed next, Fig. 5. The Cyclic displacement shown in Fig. 2, magnified by a factor of 50 are applied on node 3 in the X direction. Ex20 has stiffness-based beam-column, Ex21 has flexibility-based beam-column with element iteration, and Ex22 has flexibility-based beam-column without element iteration. For transient analysis, HHT integration scheme in Ex20 and Shing method in Ex21 are used with $\alpha = -0.1$, $\beta = 0.3025$ and $\gamma = 0.6$.

5.1 MATLAB

```

1 %-----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex20to22.m
7 %-----
8 % Description
9 % 1. Static and transient analysis
10 % 2. Load control
11 % 3. Iterative method
12 % 4. Stiffness-based 2D beam-column,
13 % flexibility-based 2D beam-column 1 and 2
14 % 5. Layer sections
15 % 6. Hardening material
16 %-----
17 % Section AnalysisType
18 % AnalysisType = 1: Displacement Control

```

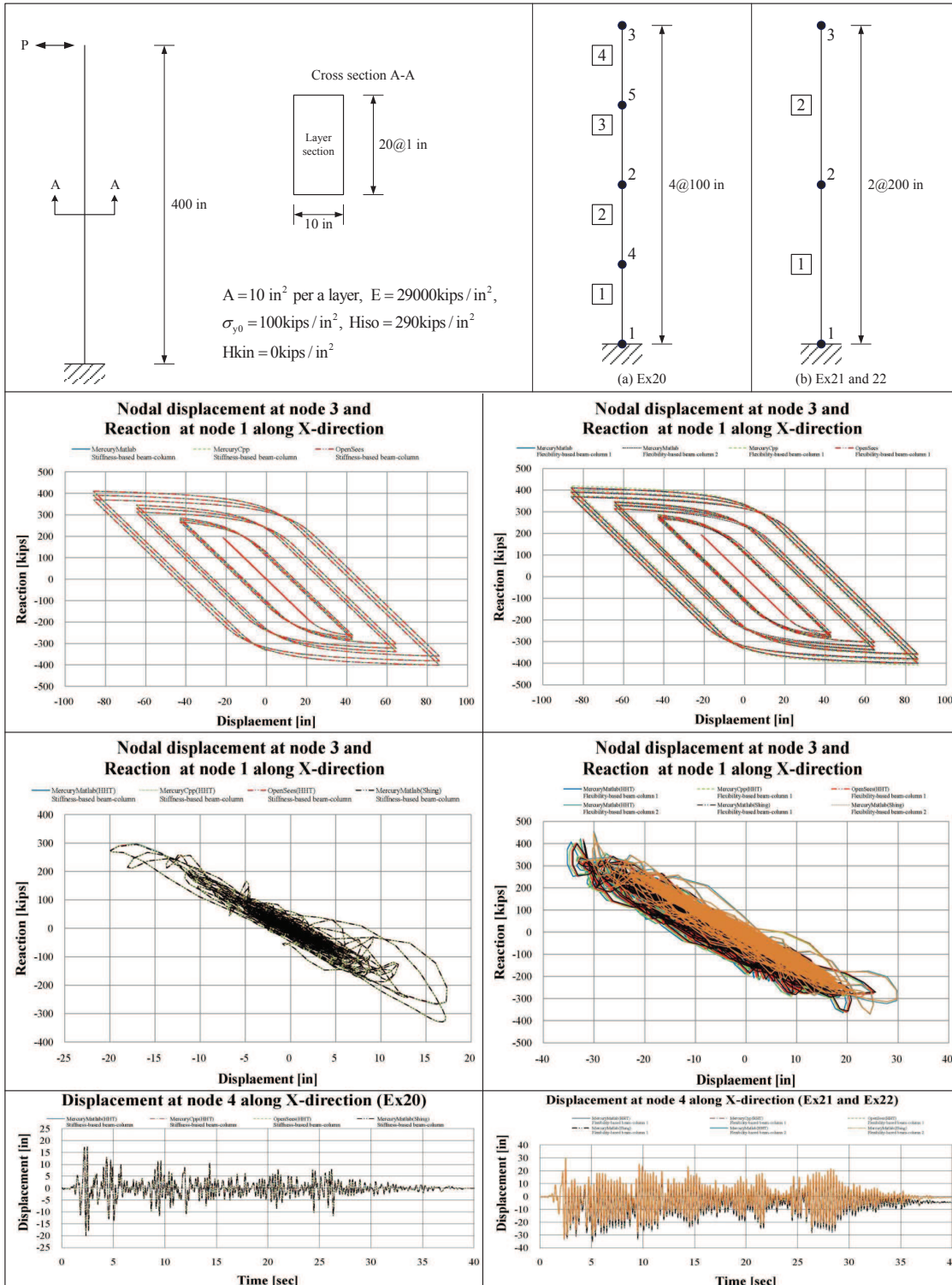


Figure 5: Examples 20- 22

```

19 % AnalysisType = 2: Transient analysis
20 AnalysisType = 2;
21 %-----
22 % Section DynType
23 % DynType = 1: Newmark beta
24 % DynType = 2: HHT
25 % DynType = 3: Shing
26 % DynType = 4: None
27 DynType = 3;
28 %-----
29 % Section EleType
30 % EleType = 1: Stiffness-based 2D beam-column (Ex20)
31 % EleType = 2: Flexibility-based 2D beam-column by Spacone (Ex21)
32 % EleType = 3: Flexibility-based 2D beam-column by Carol (Ex22)
33 EleType = 2;
34 %-----
35 % Preface
36 Unit = {'kip', 'in'};
37 StrMode = {2, 3};
38 %-----
39 % Control block
40 if ( (EleType == 1 || EleType == 2) && (DynType == 1 || DynType == 2) )
41     Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
42                             {'ModifiedNewtonRaphson', 1000, 1.0e-6, 'ForceNorm'};
43                             {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};
44                             };
45                             'transient',{ {'NewtonRaphson', 10, 1.0e-3, 'DisplNorm'};
46                                             {'ModifiedNewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
47                                             {'InitialStiffness', 1000, 1.0e-3, 'DisplNorm'};
48                                             };
49                             'element',{ {'NewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
50                                             {'ModifiedNewtonRaphson', 1000, 1.0e-3, 'DisplNorm'};
51                                             {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'}
52                                             };
53                             };
54 elseif ( (EleType == 3) && (DynType == 1 || DynType == 2) )
55     Iteration = {'static',{ {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};};
56                 'transient',{ {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'};
57                                 };};
58 end
59 if (DynType == 3)
60     Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
61                             {'ModifiedNewtonRaphson', 1000, 1.0e-6, 'ForceNorm'};
62                             {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};
63                             };
64                             'transient',{ {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'}; % for shing method
65                                             };
66                             'element',{ {'NewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
67                                             {'ModifiedNewtonRaphson', 1000, 1.0e-3, 'DisplNorm'};
68                                             {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'}
69                                             };
70                             };
71 end
72
73
74 if (DynType == 1)
75     Integration = {'Newmark', 0, 1/4, 1/2, 0, 0};
76     eigens = {0.02, 0.02};
77 end
78 if (DynType == 2)
79     Integration = {'HHT', 0, -0.1, 0.3025, 0.6, 0, 0};
80     eigens = {0.02, 0.02};
81 end
82 if (DynType == 3)
83     Integration = {'Shing', 0, -0.1, 0.3025, 0.6, 0, 0, 10};
84     eigens = {0.02, 0.02};
85 end
86 %-----
87 % Geometry block
88 if (EleType == 1)
89     nodcoord = {1, 0, 0;
90                4, 0, 100;
91                2, 0, 200;
92                5, 0, 300;
93                3, 0, 400};
94 else
95     nodcoord = {1, 0, 0;
96                2, 0, 200;
97                3, 0, 400};
98 end
99 %          nodtag, x, y, z
100 constraint = {1, 1, 1, 1};
101 %-----
102 % Element block
103 % elements = { { etetag, 'eletype', in, jn, nlp, sectag } };
104 if (EleType == 1)
105     elements = { {1, 'StiffnessBased2DBeamColumn', 1, 4, 5, 1};
106                 {2, 'StiffnessBased2DBeamColumn', 4, 2, 5, 1};
107                 {3, 'StiffnessBased2DBeamColumn', 2, 5, 5, 1};
108                 {4, 'StiffnessBased2DBeamColumn', 5, 3, 5, 1};};
109 elseif (EleType == 2)
110     elements = { {1, 'FlexibilityBased2DBeamColumn1', 1, 2, 5, 1};
111                 {2, 'FlexibilityBased2DBeamColumn1', 2, 3, 5, 1};};
112 elseif (EleType == 3)
113     elements = { {1, 'FlexibilityBased2DBeamColumn2', 1, 2, 5, 1};
114                 {2, 'FlexibilityBased2DBeamColumn2', 2, 3, 5, 1};};
115 end
116 %-----
117 % Section block
118 % b = 10, h = 20, number of layer = 10, hlayer = 2
119 area = 10; mtag = 1; count = 0;
120 for ydis = -9.5:1.0:9.5
121     count = count + 1; lay(count,1:3) = [mtag, area, ydis];
122 end
123 nlay = size(lay,1);
124 for i = 1:nlay
125     laycell{i,1}=lay(i,1); laycell{i,2}=lay(i,2); laycell{i,3}=lay(i,3);

```

```

126 end
127 % sections = { sectag, 'Layer', {mattag, A, y} }
128 sections = {1, 'Layer', laycell };
129 clear area; clear mtag; clear count; clear nlay; clear lay;
130 clear laycell; clear i; clear ydis;
131 %-----
132 % Material block
133 % mass density = 15.2 (slug/ft^3)
134 %               = 15.2 (lb*s^2/ft/ft^3)
135 %               = 15.2*(10^-3)/(12^4) (kips*s^2/in^4)
136 materials = { {1, 'Hardening', 29*10^3, 100, 290, 0, 7.3302e-007};};
137 %-----
138 % Force block
139 if (AnalysisType == 1)
140     DispInput = load('cyclicwave.txt');
141     row = size(DispInput,1);
142     for i = 1:row
143         DispCell{i} = 50*DispInput(i);
144     end
145     forces = { 1, 'Static', {'NodalForces', {3, 1, 0} };
146               2, 'DispCtrl', {3, 1, DispCell} };
147     clear DispInput; clear row; clear i; clear DispCell;
148 elseif (AnalysisType == 2)
149     ga = load('ElCentro_g_0_01_Matlab.txt');
150     nga = size(ga, 1);
151     for i = 1:nga
152         groundacceleration{i,1} = ga(i,1);
153         groundacceleration{i,2} = ga(i,2);
154         groundacceleration{i,3} = ga(i,3);
155     end
156     forces = { 1, 'Static', {'NodalForces', {3, 1, 0} };
157               2, 'Acceleration', {50*386.4, groundacceleration} };
158     clear ga; clear nga; clear i; clear groundacceleration;
159 end
160 %-----

```

5.2 C++

```

1  --- *****
2  --- AnalysisType = 1: Static analysis
3  --- AnalysisType = 2: Trnasient analysis
4  AnalysisType = 2
5  --- EleType = 1: Stiffness-based beam-column (Ex20)
6  --- EleType = 2: Flexibility-based beam-column (Ex21)
7  EleType = 1
8  --- *****
9  if (EleType == 1) then
10     nodes = {{1, 0, 0};
11              {4, 0, 100, 'mass', 0.0146604, 0.0146604, 0};
12              {2, 0, 200, 'mass', 0.0146604, 0.0146604, 0};
13              {5, 0, 300, 'mass', 0.0146604, 0.0146604, 0};
14              {3, 0, 400, 'mass', 0.0073302, 0.0073302, 0}};
15     elements = { { 1, 'StiffnessBased2DBeamColumn', 1, 4, {1, 5} };
16                  { 2, 'StiffnessBased2DBeamColumn', 4, 2, {1, 5} };
17                  { 3, 'StiffnessBased2DBeamColumn', 2, 5, {1, 5} };
18                  { 4, 'StiffnessBased2DBeamColumn', 5, 3, {1, 5} } };
19 end
20 if (EleType == 2) then
21     nodes = {{1, 0, 0};
22              {2, 0, 200, 'mass', 0.0293208, 0.0293208, 0};
23              {3, 0, 400, 'mass', 0.0146604, 0.0146604, 0}}
24     flexparams = {100000, 1e-3}
25     elements = { { 1, 'FlexibilityBased2DBeamColumn', 1, 2, {1, 5}, flexparams };
26                  { 2, 'FlexibilityBased2DBeamColumn', 2, 3, {1, 5}, flexparams } }
27 end
28 ---
29
30 sections = {
31 1, 'Fiber',
32 --- MatTag, Area, y-loc, z-loc
33 { 1, 10, -9.5, 0;
34   1, 10, -8.5, 0;
35   1, 10, -7.5, 0;
36   1, 10, -6.5, 0;
37   1, 10, -5.5, 0;
38   1, 10, -4.5, 0;
39   1, 10, -3.5, 0;
40   1, 10, -2.5, 0;
41   1, 10, -1.5, 0;
42   1, 10, -0.5, 0;
43   1, 10, 0.5, 0;
44   1, 10, 1.5, 0;
45   1, 10, 2.5, 0;
46   1, 10, 3.5, 0;
47   1, 10, 4.5, 0;
48   1, 10, 5.5, 0;
49   1, 10, 6.5, 0;
50   1, 10, 7.5, 0;
51   1, 10, 8.5, 0;
52   1, 10, 9.5, 0; } };
53 ---
54 materials = { {1, 'hardening', 29000, 0, 100, 290, 0} }
55 --- *****
56 model = StructureModel(2,3)
57 model.addNodes(nodes)
58 model.addMaterials(materials)
59 model.addSections(sections)
60 model.addElements(elements)
61 model.constrainNode(1,1,1,1)
62 if (AnalysisType == 1) then
63     model.constrainNode(3,1,0,0)
64 end
65 --- *****

```



```

66 -- Static analysis
67 if (AnalysisType == 1) then
68   print("Static analysis started\n")
69   -- *****
70   function generateincrementalload()
71     -- format:          tag          node dof
72     local loadform = {'incrementalnodaldisplacement', 3, 1}
73     local f = assert(io.open('cyclicwave.txt','r'))
74     local n = f:read("*number")
75     while (n ~= nil) do
76       table.insert(loadform, 50*n)
77       n = f:read("*number")
78     end
79     f:close()
80     return loadform
81   end
82   -- *****
83   staticloading = LoadDescription()
84   -- format: 'staticnodalload' <node> <dof> <amplitude>
85   l = generateincrementalload()
86   staticloading:addLoad(l)
87   -- *****
88   displ = {}
89   function displperstep(increment)
90     dx3,dy3,dz3 = model:nodeDisplacements(3)
91     table.insert(displ, dx3)
92   end
93   --
94   react = {}
95   function reactperstep(increment)
96     fx1,fy1,fz1 = model:nodeRestoringForces(1)
97     table.insert(react, fx1)
98   end
99   --
100  -- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
101  solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=100000})
102  analysis = StaticAnalysis(solver)
103  analysis:setStructureModel(model)
104  analysis:addcallback(displperstep,"increment")
105  analysis:addcallback(reactperstep,"increment")
106  analysis:solve(staticloading)
107  -- *****
108  -- Set output file
109  function writedata1(x, fname)
110    local f = assert(io.open(fname,'w'))
111    local writenl = 0
112    for i,v in ipairs(x) do
113      f:write(v, " ")
114      writenl = writenl + 1
115      -- length of row size: writenl
116      if (writenl > 0) then
117        writenl = 0
118        f:write("\n")
119      end
120    end
121    f:close()
122  end
123  --
124  if (EleType == 1) then
125    writedata1(displ, 'Ex20StaticNodalDisp_3.dat')
126    writedata1(react, 'Ex20StaticReact_1.dat')
127  end
128  if (EleType == 2) then
129    writedata1(displ, 'Ex21StaticNodalDisp_3.dat')
130    writedata1(react, 'Ex21StaticReact_1.dat')
131  end
132  print("Static analysis ended\n")
133 end
134 -- *****
135 if (AnalysisType == 2) then
136   print("Transient analysis started\n")
137   earthquakeloading = LoadDescription()
138   accelamp = 50*386.4
139   earthquakeloading:addLoad({'groundmotion', 'ElCentro_g_0_01_OpenSees.txt', dt=0.01', 1, accelamp})
140   -- *****
141   displ = {}
142   function displvertime(time)
143     dx3,dy3,dz3 = model:nodeDisplacements(3)
144     table.insert(displ, dx3)
145   end
146   --
147   react = {}
148   function reactvertime(time)
149     fx1,fy1,fz1 = model:nodeRestoringForces(1)
150     table.insert(react, fx1)
151   end
152   -- *****
153   -- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
154   solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=1000})
155   transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.1, 0.3025, 0.6)
156   transientanalysis:addcallback(displvertime, "timestep")
157   transientanalysis:addcallback(reactvertime, "timestep")
158   if (EleType == 1) then
159     model:setRayleighCoefficients(9.693343,0.000038)
160   end
161   if (EleType == 2) then
162     model:setRayleighCoefficients(0.757867,0.000287)
163   end
164   transientanalysis:solve(4000)
165   -- *****
166   function writedata1(x, fname)
167     local f = assert(io.open(fname,'w'))
168     local writenl = 0
169     for i,v in ipairs(x) do
170       f:write(v, " ")
171       writenl = writenl + 1
172       -- length of row size: writenl

```

```

173         if (writtenl > 0) then
174             writtenl = 0
175             f:write("\n")
176         end
177     end
178     f:close()
179 end
180
181 if (EleType == 1) then
182     writedata1(displ, 'Ex20HHTNodalDisp_3.dat')
183     writedata1(react, 'Ex20HHTReact_1.dat')
184 end
185 if (EleType == 2) then
186     writedata1(displ, 'Ex21HHTNodalDisp_3.dat')
187     writedata1(react, 'Ex21HHTReact_1.dat')
188 end
189 print("Transient analysis ended\n")
190 end
191 -----

```

6 Zero-length and Beam Column, Nonlinear Steel Element, load control

The implementation of the zero-length element is examined next in combination with stiffness-based beam-column and zero-length element, elastic and bilinear materials, Fig. 6. The incremental forces are increase by -10kN up to -50kN at node 2 in the X direction.

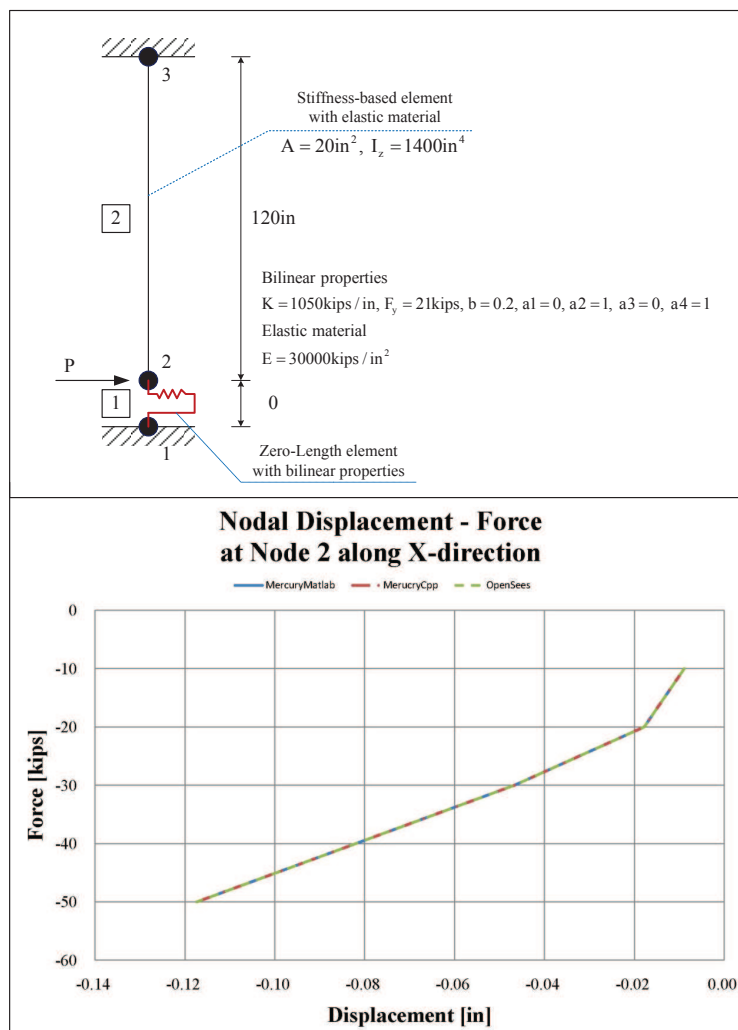


Figure 6: Examples 23

6.1 MATLAB

```

1 %=====
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex23.m
7 %=====
8 % Description
9 % 1. Static analysis
10 % 2. Static force and load control
11 % 3. Iterative method
12 % 4. Stiffness-based 2D beam column and Zero-Length 2D element
13 % 5. General section
14 % 6. Bilinear material
15 %-----
16 % Preface
17 Unit = {'kip', 'in'};
18 StrMode = {2, 3};
19 %-----
20 % Control block
21 Iteration = {'static', { {'InitialStiffness', 100, 1.0e-8, 'ForceNorm'};
22 }
23 };
24 %-----
25 % Geometry block
26 nodcoord = {1, 0, 0;
27 2, 0, 0;
28 3, 0, 120};
29 constraint = {1, 1, 1, 1;
30 3, 1, 1, 1};
31 %-----
32 % Element block
33 elements = { {1, 'ZeroLength2D', 1, 2, 0, 1, 0, deg2rad(90)}
34 {2, 'StiffnessBased2DBeamColumn', 2, 3, 1, 1}};
35 %-----
36 % Section block
37 sections = { 1, 'General', {2, 20, 0, 0, 1400} };
38 %-----
39 % Material block
40 materials = { {1, 'Bilinear', 1050, 21, 0.2, 0, 0, 1, 0, 1};
41 {2, 'Elastic', 30000, 0, 0} };
42 %-----
43 % Force block
44 forces = { 1, 'Static', {'NodalForces', {2, 1, -10}};
45 2, 'LoadCtrl', {2, 1, {-20,-30,-40,-50} } };
46 %-----

```

6.2 C++

```

1 -----
2 nodes = { {1, 0, 0};
3 {2, 0, 0};
4 {3, 0, 120} };
5 -----
6 { eleTag, 'InterfaceElement2D', inode, jnode, { {matTag, {1,0,0} } }, { {secTag}, {0,1,0},{-1,0,0} }
7 elements = { {1, 'InterfaceElement2D', 1, 2, { { 1, {1,0,0} } }, { } };
8 {2, 'StiffnessBased2DBeamColumn', 2, 3, {1, 1} } };
9 -----
10 sections = { 1, 'general', {2, 20, 1400} };
11 -----
12 materials = { {1, 'bilinear', 1050, 0, 21, 0.2, 0, 1, 0, 1};
13 {2, 'elastic', 30000, 0, 0} };
14 -----
15 model = StructureModel(2,3)
16 model.addNodes(nodes)
17 model.addMaterials(materials)
18 model.addSections(sections)
19 model.addElements(elements)
20 -----
21 model.constrainNode(1,1,1,1)
22 model.constrainNode(3,1,1,1)
23 -----
24 staticloading = LoadDescription()
25 staticloading.addLoad({'incrementalnodalload', 2, 1, -10,-20,-30,-40,-50})
26 -----
27 displ = {}
28 function displperstep(increment)
29 dx2,dy2,dz2 = model.nodeDisplacements(2)
30 table.insert(displ, dx2)
31 end
32 solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
33 analysis = StaticAnalysis(solver)
34 analysis.setStructureModel(model)
35 analysis.addcallback(displperstep,"increment")
36 analysis.solve(staticloading)
37 -----
38 == Set output file
39 function writedata1(x, fname)
40 local f = assert(io.open(fname, 'w'))
41 local writenl = 0
42 for i,v in ipairs(x) do
43 f:write(v, " ")
44 writenl = writenl + 1
45 -- length of row size: writenl
46 if (writenl > 0) then
47 writenl = 0
48 f:write("\n")
49 end
50 end
51 f:close()
52 end
53 writedata1(displ, 'Ex23NodalDisp-2.dat')

```

7 Zero-length Section, and Beam Column, Fiber, Nonlinear Steel Element, load control

The zero length section element is validated next in a similar way as in the preceding example, Fig. 7. Incremental forces of -20kN up to -400kN are applied on node 2 in the X direction.

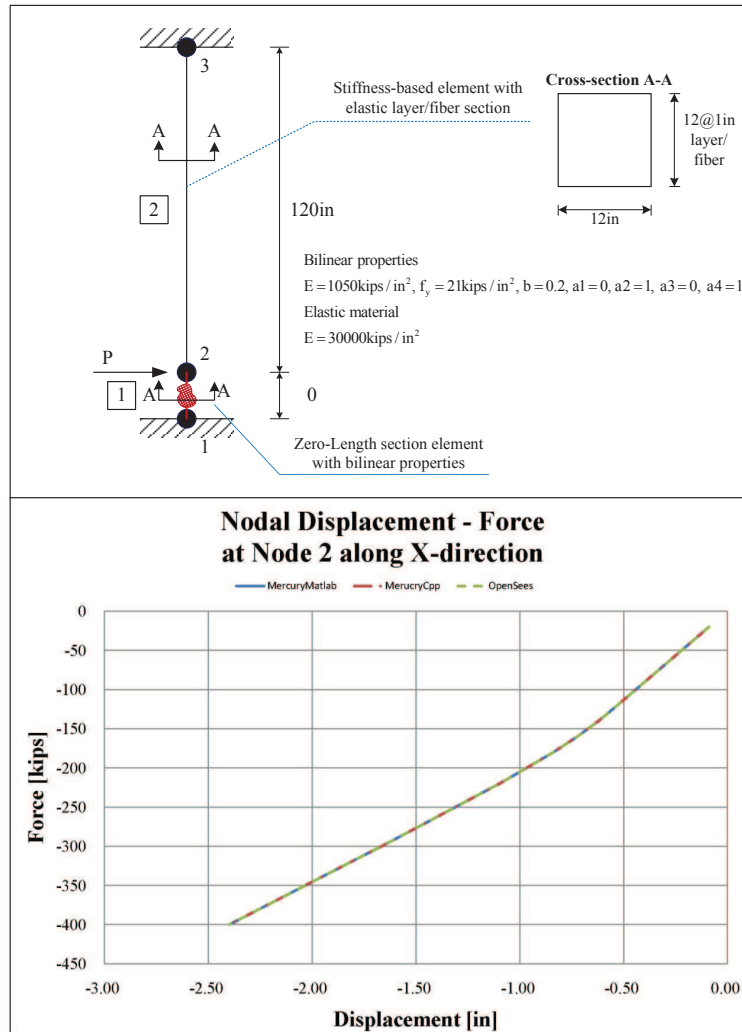


Figure 7: Examples 24

```

1 \subsection {MATLAB}
2 %
3 % Mercury Matlab Version 1.0.1
4 % Written by Dae-Hung Kang, CU-NEES
5 % Copyright 2009, CU-NEES
6 % Written : October 2009.
7 % File name: Ex24.m
8 %
9 % Description
10 % 1. Static analysis
11 % 2. Static force and load control
12 % 3. Iterative method
13 % 4. Stiffness-based 2D beam column and Zero-Length 2D section element
14 % 5. General section
15 % 6. Bilinear material
16 %
17 % Preface
18 Unit = {'kip', 'in'};
19 StrMode = {2, 3};
20 %
21 % Control block
22 Iteration = {'static', {'InitialStiffness', 100, 1.0e-8, 'ForceNorm'}};
23 };
24 %
25 %
26 % Geometry block
27 nodcoord = {1, 0, 0;

```

```

28     2, 0, 0;
29     3, 0, 120};
30 constraint = {1, 1, 1, 1;
31             3, 1, 1, 1};
32 -----
33 % Element block
34 elements = { {1, 'ZeroLength2DSection', 1, 2, deg2rad(90), 1}
35             {2, 'StiffnessBased2DBeamColumn', 2, 3, 3, 2}};
36 -----
37 % Section block
38 sections = { 2, 'Layer', {2, 12, 5.5;
39                        2, 12, 4.5;
40                        2, 12, 3.5;
41                        2, 12, 2.5;
42                        2, 12, 1.5;
43                        2, 12, 0.5;
44                        2, 12, -0.5;
45                        2, 12, -1.5;
46                        2, 12, -2.5;
47                        2, 12, -3.5;
48                        2, 12, -4.5;
49                        2, 12, -5.5}};
50     1, 'Layer', {1, 12, 5.5;
51                1, 12, 4.5;
52                1, 12, 3.5;
53                1, 12, 2.5;
54                1, 12, 1.5;
55                1, 12, 0.5;
56                1, 12, -0.5;
57                1, 12, -1.5;
58                1, 12, -2.5;
59                1, 12, -3.5;
60                1, 12, -4.5;
61                1, 12, -5.5} };
62 -----
63 % Material block
64 materials = { {1, 'Bilinear', 1050, 21, 0.2, 0, 0, 1, 0, 1};
65             {2, 'Elastic', 30000, 0, 0} };
66 -----
67 % Force block
68 forces = { 1, 'Static', {'NodalForces', {2, 1, -20}};
69           2, 'LoadCtrl', {2, 1, {-40, -60, -80, -100, -120, ...
70                               -140, -160, -180, -200, -220, ...
71                               -240, -260, -280, -300, -320, ...
72                               -340, -360, -380, -400} } };
73 -----

```

7.1 C++

```

1  --- *****
2  nodes = { {1, 0, 0};
3            {2, 0, 0};
4            {3, 0, 120} };
5  --- *****
6  { eleTag, 'InterfaceElement2D', inode, jnode, { {matTag, {1,0,0} } }, { {secTag}}, {0,1,0},{-1,0,0} }
7  elements = { {1, 'InterfaceElement2D', 1, 2, {}, {{1}}, {0,1,0},{-1,0,0} };
8              {2, 'StiffnessBased2DBeamColumn', 2, 3, {2, 3} } };
9  --- *****
10 sections = {1, 'Fiber', {1, 12, 5.5, 0;
11                      1, 12, 4.5, 0;
12                      1, 12, 3.5, 0;
13                      1, 12, 2.5, 0;
14                      1, 12, 1.5, 0;
15                      1, 12, 0.5, 0;
16                      1, 12, -0.5, 0;
17                      1, 12, -1.5, 0;
18                      1, 12, -2.5, 0;
19                      1, 12, -3.5, 0;
20                      1, 12, -4.5, 0;
21                      1, 12, -5.5, 0};
22           2, 'Fiber', {2, 12, 5.5, 0;
23                      2, 12, 4.5, 0;
24                      2, 12, 3.5, 0;
25                      2, 12, 2.5, 0;
26                      2, 12, 1.5, 0;
27                      2, 12, 0.5, 0;
28                      2, 12, -0.5, 0;
29                      2, 12, -1.5, 0;
30                      2, 12, -2.5, 0;
31                      2, 12, -3.5, 0;
32                      2, 12, -4.5, 0;
33                      2, 12, -5.5, 0};
34           };
35  --- *****
36 materials = { {1, 'bilinear', 1050, 0, 21, 0.2, 0, 1, 0, 1};
37             {2, 'elastic', 30000, 0, 0} };
38  --- *****
39 model = StructureModel(2,3)
40 model.addNodes(nodes)
41 model.addMaterials(materials)
42 model.addSections(sections)
43 model.addElements(elements)
44  --- *****
45 model.constrainNode(1,1,1)
46 model.constrainNode(3,1,1)
47  --- *****
48 staticloading = LoadDescription()
49 staticloading.addLoad({'incrementalnodalload', 2, 1, -20,-40,-60,-80,-100,-120,-140,-160,-180,-200,-220,-240,-260,-280,-300,-320,-340})
50  --- *****
51 displ = {}
52 function displperstep(increment)
53     dx2,dy2,dz2 = model.nodeDisplacements(2)
54     table.insert(displ, dx2)

```

```

55 end
56 solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
57 analysis = StaticAnalysis(solver)
58 analysis:setStructureModel(model)
59 analysis:addcallback(displperstep,"increment")
60 analysis:solve(staticloading)
61 --- *****
62 --- Set output file
63 function writedata1(x, fname)
64     local f = assert(io.open(fname, 'w'))
65     local writenl = 0
66     for i,v in ipairs(x) do
67         f:write(v, " ")
68         writenl = writenl + 1
69         --- length of row size: writenl
70         if (writenl > 0) then
71             writenl = 0
72             f:write("\n")
73         end
74     end
75     f:close()
76 end
77 writedata1(displ, 'Ex24NodalDisp_2.dat')

```

8 Beam-column, Fiber Section, Nonlinear Material, multi-d.o.f.s displacement control, Pushover Analysis

Validation of displacement control at multiple free degrees of freedom, as described in ?? is performed next. Layered sections beam-column elements with hardening material are used, Fig. 8. Cyclic displacements shown in Fig. 2 are applied at node 2 and 3 magnified by a factor of -30 and 50 respectively. Ex25 has stiffness-based beam-column and Ex26 has flexibility-based beam-column.

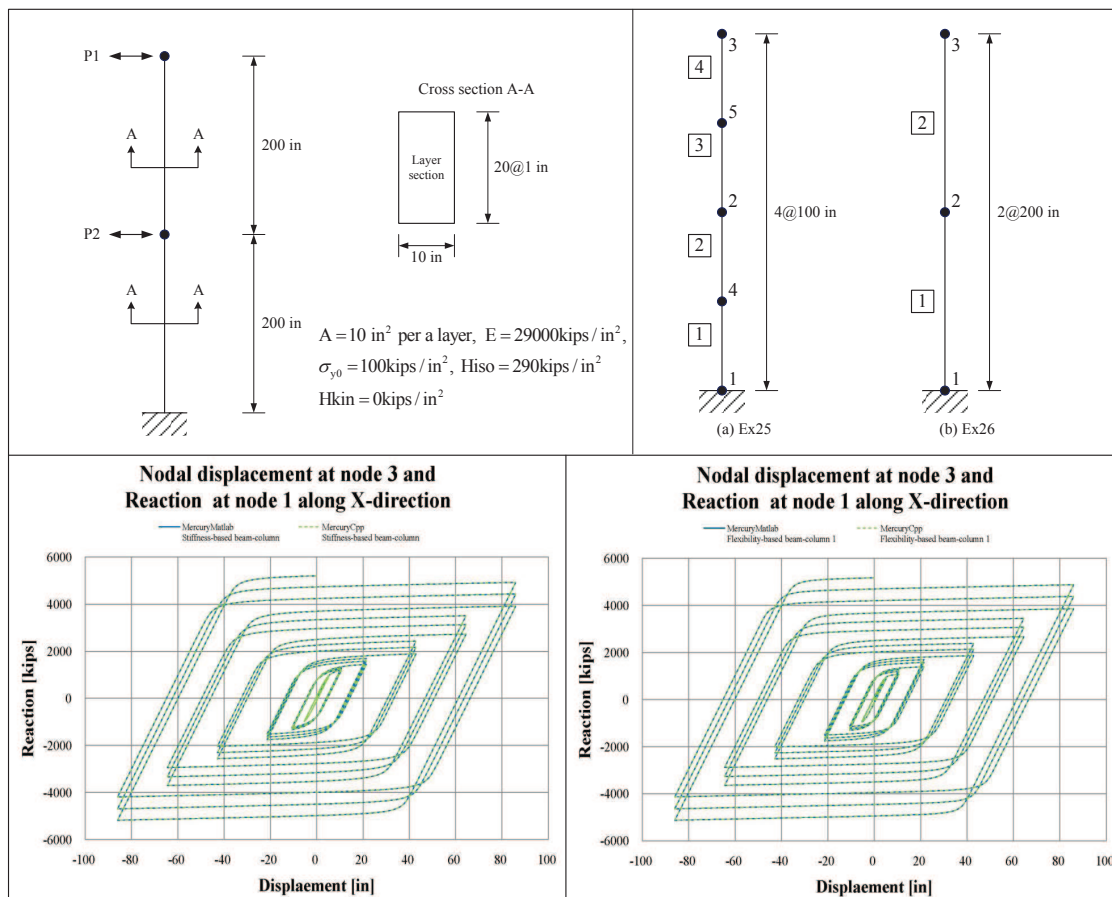


Figure 8: Examples 25- 26

8.1 MATLAB

```

1 %=====
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex25to26.m
7 %=====
8 % Description
9 % 1. Static analysis
10 % 2. Multiple displacement control
11 % 3. Iterative method
12 % 4. Stiffness-based 2D beam-column,
13 % flexibility-based 2D beam-column 1 and 2
14 % 5. Layer sections
15 % 6. Hardening material
16 %-----
17 % Section EleType
18 % EleType = 1: Stiffness-based 2D beam-column (Ex25)
19 % EleType = 2: Flexibility-based 2D beam-column by Spacone (Ex26)
20 EleType = 2;
21 %-----
22 % Preface
23 Unit = {'kip', 'in'};
24 StrMode = {2, 3};
25 %-----
26 % Control block
27 Iteration = {'static', { {'NewtonRaphson', 100, 1.0e-8, 'ForceNorm'};
28 {'ModifiedNewtonRaphson', 1000, 1.0e-8, 'ForceNorm'};
29 {'InitialStiffness', 10000, 1.0e-8, 'ForceNorm'};
30 };
31 'element', { {'NewtonRaphson', 100, 1.0e-5, 'DisplNorm'};
32 {'ModifiedNewtonRaphson', 1000, 1.0e-5, 'DisplNorm'};
33 {'InitialStiffness', 100000, 1.0e-5, 'DisplNorm'}
34 };
35 };
36 %-----
37 % Geometry block
38 if (EleType == 1)
39 nodcoord = {1, 0, 0;
40 4, 0, 100;
41 2, 0, 200;
42 5, 0, 300;
43 3, 0, 400};
44 elseif (EleType == 2)
45 nodcoord = {1, 0, 0;
46 2, 0, 200;
47 3, 0, 400};
48 end
49 % nodtag, x, y, z
50 constraint = {1, 1, 1, 1};
51 %-----
52 % Element block
53 elements = { eletag, 'eletype', in, jn, nlp, sectag };
54 if (EleType == 1)
55 elements = { {1, 'StiffnessBased2DBeamColumn', 1, 4, 5, 1};
56 {2, 'StiffnessBased2DBeamColumn', 4, 2, 5, 1};
57 {3, 'StiffnessBased2DBeamColumn', 2, 5, 5, 1};
58 {4, 'StiffnessBased2DBeamColumn', 5, 3, 5, 1}};
59 elseif (EleType == 2)
60 elements = { {1, 'FlexibilityBased2DBeamColumn1', 1, 2, 5, 1};
61 {2, 'FlexibilityBased2DBeamColumn1', 2, 3, 5, 1}};
62 end
63 %-----
64 % Section block
65 % b = 10, h = 20, number of layer = 10, hlayer = 2
66 area = 10; mtag = 1; count = 0;
67 for ydis = -9.5:1.0:9.5
68 count = count + 1; lay(count,1:3) = [mtag, area, ydis];
69 end
70 nlay = size(lay,1);
71 for i = 1:nlay
72 laycell{i,1}=lay(i,1); laycell{i,2}=lay(i,2); laycell{i,3}=lay(i,3);
73 end
74 % sections = { sectag, 'Layer', {mattag, A, y} }
75 sections = {1, 'Layer', laycell};
76 clear area; clear mtag; clear count; clear nlay; clear lay;
77 clear laycell; clear i; clear ydis;
78 %-----
79 % Material block
80 % mass density = 15.2 (slug/ft^3)
81 % = 15.2 (lb*s^2/ft/ft^3)
82 % = 15.2*(10^-3)/(12^4) (kips*s^2/in^4)
83 materials = { {1, 'Hardening', 29*10^3, 100, 290, 0, 7.3302e-007}};
84 %-----
85 % Force block
86 DispInput = load('cyclicwave.txt');
87 row = size(DispInput,1);
88 for i = 1:row
89 DispCell1{i} = -30*DispInput(i);
90 DispCell2{i} = 50*DispInput(i);
91 end
92 forces = { 1, 'Static', {'NodalForces', {3, 1, 0}};
93 2, 'DispCtrl', {2, 1, DispCell1;
94 3, 1, DispCell2}};
95 clear DispInput; clear row; clear i; clear DispCell1; clear DispCell2;
96 %-----

```

8.2 C++

```

1 --- *****
2 --- EleType = 1: Stiffness-based beam-column (Ex25)
3 --- EleType = 2: Flexibility-based beam-column (Ex26)

```

```

4 EleType = 2
5 --- *****
6 if (EleType == 1) then
7     nodes = {{1, 0, 0},
8              {4, 0, 100},
9              {2, 0, 200},
10             {5, 0, 300},
11             {3, 0, 400}};
12     elements = { { 1, 'StiffnessBased2DBeamColumn', 1, 4, {1, 5} };
13                 { 2, 'StiffnessBased2DBeamColumn', 4, 2, {1, 5} };
14                 { 3, 'StiffnessBased2DBeamColumn', 2, 5, {1, 5} };
15                 { 4, 'StiffnessBased2DBeamColumn', 5, 3, {1, 5} } };
16 end
17
18 if (EleType == 2) then
19     nodes = {{1, 0, 0},
20             {2, 0, 200},
21             {3, 0, 400}};
22     flexparams = {1000, 1e-5};
23     elements = { { 1, 'FlexibilityBased2DBeamColumn', 1, 2, {1, .5}, flexparams };
24                 { 2, 'FlexibilityBased2DBeamColumn', 2, 3, {1, .5}, flexparams } };
25 end
26 --- *****
27 sections = {
28     1, 'Fiber',
29     --- MatTag, Area, y-loc, z-loc
30     { 1, 10, -9.5, 0;
31       1, 10, -8.5, 0;
32       1, 10, -7.5, 0;
33       1, 10, -6.5, 0;
34       1, 10, -5.5, 0;
35       1, 10, -4.5, 0;
36       1, 10, -3.5, 0;
37       1, 10, -2.5, 0;
38       1, 10, -1.5, 0;
39       1, 10, -0.5, 0;
40       1, 10, 0.5, 0;
41       1, 10, 1.5, 0;
42       1, 10, 2.5, 0;
43       1, 10, 3.5, 0;
44       1, 10, 4.5, 0;
45       1, 10, 5.5, 0;
46       1, 10, 6.5, 0;
47       1, 10, 7.5, 0;
48       1, 10, 8.5, 0;
49       1, 10, 9.5, 0; } };
50 --- *****
51 materials = { {1, 'hardening', 29000, 0, 100, 290, 0} }
52 --- *****
53 model = StructureModel(2,3)
54 model:addNodes(nodes)
55 model:addMaterials(materials)
56 model:addSections(sections)
57 model:addElements(elements)
58 model:constrainNode(1,1,1,1)
59 model:constrainNode(2,1,0,0)
60 model:constrainNode(3,1,0,0)
61 --- *****
62 --- Static analysis
63 function generateincrementalload2()
64     --- format:          tag          node dof
65     local loadform = {'incrementalnodaldisplacement', 2, , 1}
66     local f = assert(io.open('cyclicwave.txt','r'))
67     local n = f:read("*number")
68     while (n ~= nil) do
69         table.insert(loadform, -30*n)
70         n = f:read("*number")
71     end
72     f:close()
73     return loadform
74 end
75 function generateincrementalload3()
76     --- format:          tag          node dof
77     local loadform = {'incrementalnodaldisplacement', 3, , 1}
78     local f = assert(io.open('cyclicwave.txt','r'))
79     local n = f:read("*number")
80     while (n ~= nil) do
81         table.insert(loadform, 50*n)
82         n = f:read("*number")
83     end
84     f:close()
85     return loadform
86 end
87 --- *****
88 staticloading = LoadDescription()
89 --- format: 'staticnodalload' <node> <dof> <amplitude>
90 l2 = generateincrementalload2()
91 l3 = generateincrementalload3()
92 staticloading:addLoad(l2)
93 staticloading:addLoad(l3)
94 --- *****
95 displ = {}
96 function displperstep(increment)
97     dx2,dy2,dz2 = model:nodeDisplacements(2)
98     dx3,dy3,dz3 = model:nodeDisplacements(3)
99     table.insert(displ, dx2)
100    table.insert(displ, dx3)
101 end
102 ---
103 react = {}
104 function reactperstep(increment)
105     fx1,fy1,fz1 = model:nodeRestoringForces(1)
106     print("Work\n");
107     table.insert(react, fx1)
108 end
109 --- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
110 --- solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-5, iterations=10000})

```



```

111 --- multisolver <type> <disp delta> <residual> <max iterations>
112 solver = NonlinearSolver(" multisolver",
113                          "newtonraphson", 1e-5, 1e-5, 100,
114                          "initialstiffness", 1e-5, 1e-5, 1000)
115
116
117
118 analysis = StaticAnalysis(solver)
119 analysis:setStructureModel(model)
120 analysis:addcallback(displperstep,"increment")
121 analysis:addcallback(reactperstep,"increment")
122 analysis:solve(staticloading)
123 --- *****
124 --- Set output file
125 function writedata1(x, fname)
126   local f = assert(io.open(fname, 'w'))
127   local writenl = 0
128   for i,v in ipairs(x) do
129     f:write(v, " ")
130     writenl = writenl + 1
131     --- length of row size: writenl
132     if (writenl > 0) then
133       writenl = 0
134       f:write("\n")
135     end
136   end
137   f:close()
138 end
139 function writedata2(x, fname)
140   local f = assert(io.open(fname, 'w'))
141   local writenl = 0
142   for i,v in ipairs(x) do
143     f:write(v, " ")
144     writenl = writenl + 1
145     --- length of row size: writenl
146     if (writenl > 1) then
147       writenl = 0
148       f:write("\n")
149     end
150   end
151   f:close()
152 end
153 ---
154 if (EleType == 1) then
155   writedata2(displ, 'Ex25NodalDisp_2_3.dat')
156   writedata1(react, 'Ex25React_1.dat')
157 end
158 if (EleType == 2) then
159   writedata2(displ, 'Ex26NodalDisp_2_3.dat')
160   writedata1(react, 'Ex26React_1.dat')
161 end

```

9 Reinforced Concrete Beam-Column, Fiber Section, Transient Analysis

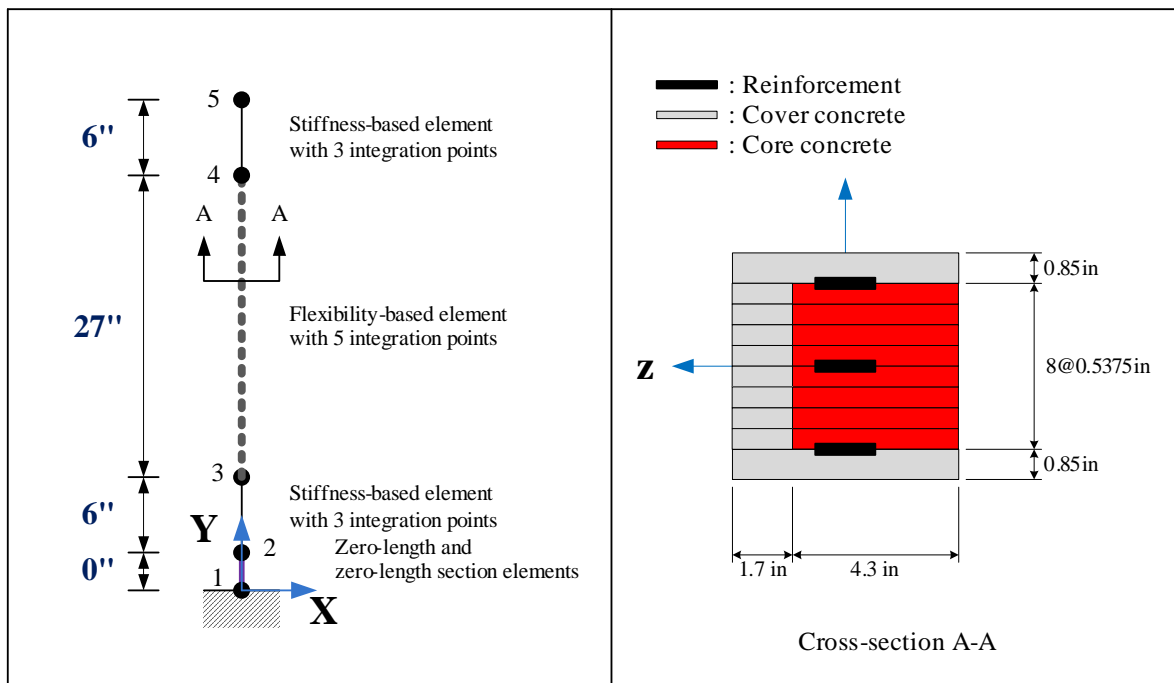


Figure 9: Beam-column elements for Ex27 and Ex28

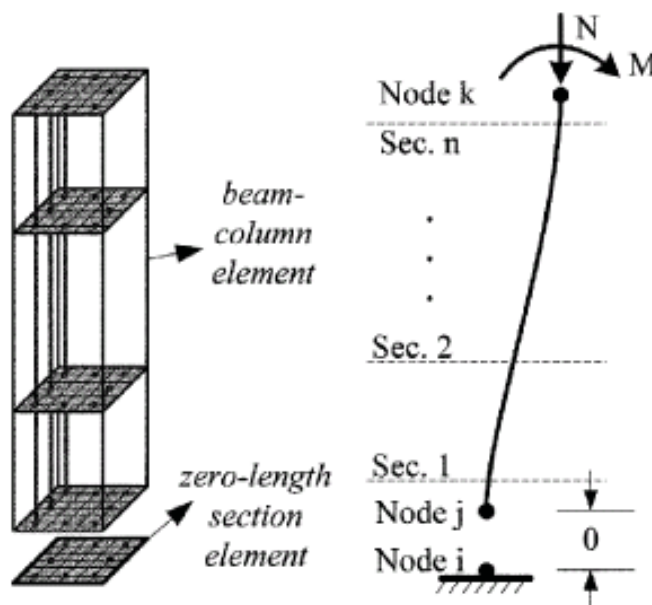


Figure 10: Bar slip zero-length fiber section element (?)

This example consists of four types of beam-column element with layer section and material constitutive models for reinforced concrete, Fig. 9. In Ex27, material constitutive model of confined (core) or unconfined (cover) concrete using the modified Kent-Park model, whereas in Ex28, material constitutive model of confined (core) or unconfined (cover) concrete based on the anisotropic damage model with permanent strain are used. Steel is modeled with the modified Giuffre-Monegotto-Pinto model, and the shear spring in the zero-length element has a bilinear model.

At the base, a zero-length section and zero-length element are used to capture bond-slip. Fig. 10 succinctly describes the bar-slip zero-length fiber-section element (?) used. ? uses an empirically derived stress-deformation relation for the bar-slip fiber material between concrete and reinforcement. The material properties of the concrete are the same as those in the adjacent column element except the residual stress at large strains is taken as $0.8 \cdot \sigma'_c$ (?). Zero-length elements are used to account for shear-deformations using elastic spring elements at both ends of beams and columns as well. The joints are assumed to be rigid otherwise.

Table 2 to 5 describe material properties of Ex27 and Ex28. Applied masses from node 2 to node 5 are $2.50712E - 5$, $1.37891E - 4$, $1.37891E - 4$, $(2.50712E - 5 + 9.0/386.4)kips/in^3$ respectively. There are no rotation masses.

Table 2: Material properties of concrete for Ex27 (unit: kips, in)

Element	Concrete	E_{ts}	σ_c	ϵ_c	σ_{cu}	ϵ_{cu}	λ	σ_t
Beam-column	Cover	549.231	-3.57	-0.0026	-1.19	-0.0078	0.3	0.448
	Core	549.451	-7.5	-0.00546	-7.35	-0.01638	0.3	0.650
Zero-length section	Cover	105.000	-3.57	-0.0136	-1.19	-0.0408	0.3	0.448
	Core	104.167	-7.5	-0.0288	-6.75	-0.0864	0.3	0.650

Table 3: Material properties of concrete for Ex28 (unit: kips, in)

Element	Concrete	E	ν	κ_0	A	k_h	k_d	K_h	K_d	D_c
Beam-column	Cover	2829	0.2	5.855E-08	1870	0.003248	0.05168	3.889E+10	22.27	1.00
	Core	1804	0.2	6.483E-06	503.6	3.125E-07	0.4054	3.159E+09	102	1.00
Zero-length section	Cover	569.3	0.2	1.689E-19	371.8	0.832	1.461	9.529E+13	100.4	1.00
	Core	396.7	0.2	0.0008326	124.8	2.084E-07	1.61	8.856E+11	62.48	1.00

Fig. 11 describes results of Ex27 and Ex28.

9.1 MATLAB

```
1 %
2 % Mercury Matlab Version 1.0.1
```

Table 4: Material properties of reinforcement for Ex27 and Ex28 (unit: kips, in)

Element	E	σ_y	b	$R0$	$cR1$	$cR2$	$a1$	$a2$	$a3$	$a4$
Beam-column element	26500	87.5	0.01	15	0.925	0.15	0	55	0	55
Zero-length section element	6949	87.5	0.01	15	0.925	0.15	0	55	0	55

Table 5: Property of shear spring in zero-length element for Ex27 and Ex28 (unit: kips, in)

Element	E	σ_y	b	$a1$	$a2$	$a3$	$a4$
Shear spring in zero-length	1690	78.2	0.173	0	55	0	55

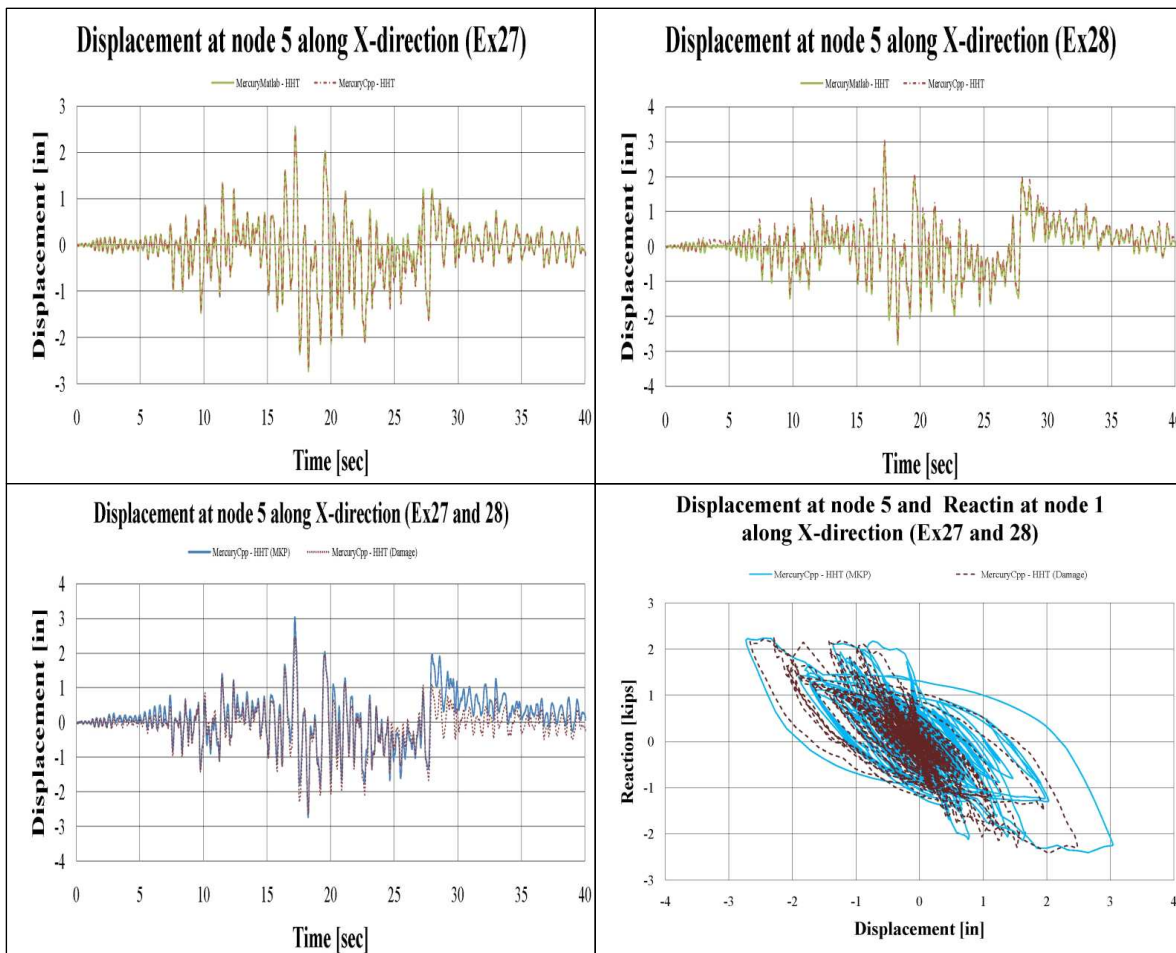


Figure 11: Output for Ex27 and Ex28

```

3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex27.m (1 column)
7
8 % Preface
9 Unit = {'kip', 'in'};
10 % ndim, ndofpn
11 StrMode = {2, 3};
12
13 % Control block
14 Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
15                       {'InitialStiffness', 10000, 1.0e-6, 'DisplNorm'};
16                       }
17           'element',{ {'NewtonRaphson', 1000, 1.0e-6, 'DisplNorm'};
18                       {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
19                       }
20           'transient',{ {'NewtonRaphson', 100, 1.0e-6, 'DisplNorm'};
21                          {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
22                          }
23           };
24 %Integration = {'HHT', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252};
25 Integration = {'Shing', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252,10};
26 addMass = {1, 0, 0, 0;
27            2, 2.50712E-05, 2.50712E-05, 0;
28            3, 0.000137891, 0.000137891, 0;
29            4, 0.000137891, 0.000137891, 0;
30            5, 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0};
31
32 nodcoord = {1, 0, 0;
33            2, 0, 0;
34            3, 0, 6;
35            4, 0, 33;
36            5, 0, 39};
37 constraint = {1, 1, 1, 1};
38
39
40 elements = { {1, 'ZeroLength2D', 1, 2, 0, 26, 0, pi()/2};
41             {2, 'ZeroLength2DSection', 1, 2, pi()/2, 3};
42             {3, 'StiffnessBased2DBeamColumn', 2, 3, 3, 1};
43             {4, 'FlexibilityBased2DBeamColumn1', 3, 4, 5, 1};
44             {5, 'StiffnessBased2DBeamColumn', 4, 5, 3, 1}};
45
46 sections = {1, 'Layer', {1, 5.1, 2.575;
47                        1, 5.1, -2.575;
48                        1, 0.91375, 1.88125;
49                        1, 0.91375, 1.34375;
50                        1, 0.91375, 0.80625;
51                        1, 0.91375, 0.26875;
52                        1, 0.91375, -0.26875;
53                        1, 0.91375, -0.80625;
54                        1, 0.91375, -1.34375;
55                        1, 0.91375, -1.88125;
56                        2, 2.31125, 1.88125;
57                        2, 2.31125, 1.34375;
58                        2, 2.31125, 0.80625;
59                        2, 2.31125, 0.26875;
60                        2, 2.31125, -0.26875;
61                        2, 2.31125, -0.80625;
62                        2, 2.31125, -1.34375;
63                        2, 2.31125, -1.88125;
64                        17, 0.147, 2.15;
65                        17, 0.098, 0;
66                        17, 0.147, -2.15};
67            3, 'Layer', {5, 5.1, 2.575;
68                       5, 5.1, -2.575;
69                       5, 0.91375, 1.88125;
70                       5, 0.91375, 1.34375;
71                       5, 0.91375, 0.80625;
72                       5, 0.91375, 0.26875;
73                       5, 0.91375, -0.26875;
74                       5, 0.91375, -0.80625;
75                       5, 0.91375, -1.34375;
76                       5, 0.91375, -1.88125;
77                       6, 2.31125, 1.88125;
78                       6, 2.31125, 1.34375;
79                       6, 2.31125, 0.80625;
80                       6, 2.31125, 0.26875;
81                       6, 2.31125, -0.26875;
82                       6, 2.31125, -0.80625;
83                       6, 2.31125, -1.34375;
84                       6, 2.31125, -1.88125;
85                       19, 0.147, 2.15;
86                       19, 0.098, 0;
87                       19, 0.147, -2.15}};
88
89 materials = { {1, 'ModKP', -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077, 549.2307692, 0 };
90             {2, 'ModKP', -7.5, -0.00546, -7.35, -0.01638, 0.3, 0.649519053, 549.4505495, 0};
91             { 5, 'ModKP', -3.57, -0.0136, -1.19, -0.0408, 0.3, 0.448121077, 105, 0};
92             { 6, 'ModKP', -7.5, -0.0288, -6.75, -0.0864, 0.3, 0.649519053, 104.1666667, 0};
93             { 17, 'ModGMP', 26500, 87.5, 0.01, 15, 0.925, 0.15, 0, 0.55, 0.55};
94             { 19, 'ModGMP', 6949, 87.5, 0.01, 15, 0.925, 0.15, 0, 0.55, 0.55};
95             { 26, 'Bilinear', 1690, 78.2, 0.173, 0, 0.55, 0.55};
96
97 % Force block
98 ga = load('NR_g_dt_0_01_Matlab.txt');
99 nga = size(ga, 1);
100 for i = 1:nga
101     groundacceleration{i,1} = ga(i,1);
102     groundacceleration{i,2} = ga(i,2);
103     groundacceleration{i,3} = ga(i,3);
104 end
105 forces = { 1, 'Static', {'NodalForces', {5, 1, 0}};
106           2, 'Acceleration', {386.4, groundacceleration}};

```

9.2 C++

```

1  --- *****
2  --- 1 column
3  --- *****
4  --- o (39)
5  --- | Stiffness-based beam-column with 2 integration points
6  --- o (33)
7  --- | Flexibility-based beam-column with 5 integration points
8  --- |
9  --- |
10 --- o (6)
11 --- | Stiffness-based beam-column with 2 integration points
12 --- o (0)
13 --- Zero-Length and zero-Length section elements (bottom bar slip and shear deformation)
14 --- -o- (0) (Fixed support)
15 --- *****
16 elements = {}
17
18 --- create ductile column node coordinates
19 --- create nodes
20
21 nodes = {{1, 0, 0, 'mass', 0, 0, 0};
22           {2, 0, 0, 'mass', 2.50712E-05, 2.50712E-05, 0};
23           {3, 0, 6, 'mass', 0.000137891, 0.000137891, 0};
24           {4, 0, 33, 'mass', 0.000137891, 0.000137891, 0};
25           {5, 0, 39, 'mass', 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0} }
26
27 barslipsectionf = 'BSColDFSection'
28 barslipspringf = 'BSColDFSS'
29 columnsection = 'ColDSection'
30 columnrigidsection = 'ColRigidSection'
31 nIp_stif = 3;
32 nIp_flex = 5;
33 flexparams = {10000, 1e-6}
34 --- Define elements
35 barslipbottom = { 1, 'InterfaceElement2D', 1, 2, {{ barslipspringf, {1,0,0} }},
36                 {{ barslipsectionf }, {0,1,0}, {-1,0,0} }
37 plasticcolumn1 = { 2, 'StiffnessBased2DBeamColumn', 2, 3, {columnsection, nIp_stif} }
38 flexcolumn = { 3, 'FlexibilityBased2DBeamColumn', 3, 4, {columnsection, nIp_flex, flexparams} }
39 plasticcolumn2 = { 4, 'StiffnessBased2DBeamColumn', 4, 5, {columnsection, nIp_stif} }
40 table.insert(elements, barslipbottom)
41 table.insert(elements, plasticcolumn1)
42 table.insert(elements, flexcolumn)
43 table.insert(elements, plasticcolumn2)
44 --- *****
45 --- Set section properties.
46
47 allsections = {
48
49 'ColDSection', 'Fiber',
50 --- Tag, Area, y-loc, z-loc
51 { 'ColDCover', 5.1, 2.575, 0,
52   'ColDCover', 5.1, -2.575, 0,
53   'ColDCover', 0.91375, 1.88125, 0,
54   'ColDCover', 0.91375, 1.34375, 0,
55   'ColDCover', 0.91375, 0.80625, 0,
56   'ColDCover', 0.91375, 0.26875, 0,
57   'ColDCover', 0.91375, -0.26875, 0,
58   'ColDCover', 0.91375, -0.80625, 0,
59   'ColDCover', 0.91375, -1.34375, 0,
60   'ColDCover', 0.91375, -1.88125, 0,
61   'ColDCore', 2.31125, 1.88125, 0,
62   'ColDCore', 2.31125, 1.34375, 0,
63   'ColDCore', 2.31125, 0.80625, 0,
64   'ColDCore', 2.31125, 0.26875, 0,
65   'ColDCore', 2.31125, -0.26875, 0,
66   'ColDCore', 2.31125, -0.80625, 0,
67   'ColDCore', 2.31125, -1.34375, 0,
68   'ColDCore', 2.31125, -1.88125, 0,
69   'ColDSteel', 0.147, 2.15, 0,
70   'ColDSteel', 0.098, 0, 0,
71   'ColDSteel', 0.147, -2.15, 0 };
72
73 'BSColDFSection', 'Fiber',
74 { 'BSColDFCover', 5.1, 2.575, 0,
75   'BSColDFCover', 5.1, -2.575, 0,
76   'BSColDFCover', 0.91375, 1.88125, 0,
77   'BSColDFCover', 0.91375, 1.34375, 0,
78   'BSColDFCover', 0.91375, 0.80625, 0,
79   'BSColDFCover', 0.91375, 0.26875, 0,
80   'BSColDFCover', 0.91375, -0.26875, 0,
81   'BSColDFCover', 0.91375, -0.80625, 0,
82   'BSColDFCover', 0.91375, -1.34375, 0,
83   'BSColDFCover', 0.91375, -1.88125, 0,
84   'BSColDFCore', 2.31125, 1.88125, 0,
85   'BSColDFCore', 2.31125, 1.34375, 0,
86   'BSColDFCore', 2.31125, 0.80625, 0,
87   'BSColDFCore', 2.31125, 0.26875, 0,
88   'BSColDFCore', 2.31125, -0.26875, 0,
89   'BSColDFCore', 2.31125, -0.80625, 0,
90   'BSColDFCore', 2.31125, -1.34375, 0,
91   'BSColDFCore', 2.31125, -1.88125, 0,
92   'BSColDFSteel', 0.147, 2.15, 0,
93   'BSColDFSteel', 0.098, 0, 0,
94   'BSColDFSteel', 0.147, -2.15, 0 };
95 --- *****
96 --- Set material properties.
97 concretemat = 'ConcreteLinearTensionSoftening'
98 steelmat = 'GiuffreMenegottoPinto'
99 sspringmat = 'Bilinear'
100 materials = {
101 {'ColDCover', concretemat, 549.2307692, 0, -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077},
102 {'ColDCore', concretemat, 549.4505495, 0, -7.5, -0.00546, -7.35, -0.01638, 0.3, 0.649519053},
103 {'BSColDFCover', concretemat, 105, 0, -3.57, -0.0136, -1.19, -0.0408, 0.3, 0.448121077},
104 {'BSColDFCore', concretemat, 104.1666667, 0, -7.5, -0.0288, -6.75, -0.0864, 0.3, 0.649519053},

```

```

105 {'ColDSteel' , steelmat, 26500, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
106 {'BSColDFSteel' , steelmat, 6949 , 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
107 {'BSColDFSS' , sspringmat, 1690, 0, 78.2, 0.173, 0, 55, 0, 55},
108 }
109 --- *****
110 function dumptable(x)
111     local result = '{'
112     for i,v in ipairs(x) do
113         if (type(v) == 'table') then
114             result = result .. dumptable(v)
115         else
116             result = result .. v .. ','
117         end
118     end
119     return result .. '}'\n'
120 end
121 ---
122 --print(dumptable(nodes))
123 --print(dumptable(elements))
124 --print(dumptable(materials))
125 --print(dumptable(allsections))
126 --- *****
127 --- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
128 model = StructureModel(2,3)
129 --- Assign all input data to Mercury
130 model:addNodes(nodes)
131 model:addMaterials(materials)
132 model:addSections(allsections)
133 model:addElements(elements)
134 --- *****
135 --- constrain bottom nodes
136 model:constrainNode(1,1,1,1)
137 --- *****
138 print("Transient analysis started\n")
139 earthquakeloading = LoadDescription()
140 earthquakeloading:addLoad({'groundmotion', 'NR-g-dt-0-01-OpneSees.txt', dt=0.01', 1, 386.4})
141 --- *****
142 displ = {}
143 function displpertime(time)
144     dx5,dy5,dz5 = model:nodeDisplacements(5)
145     table.insert(displ, dx5)
146     print("Work\n");
147 end
148 ---
149 react = {}
150 function reactpertime(time)
151     fx1,fy1 = model:nodeRestoringForces(1)
152     table.insert(react, fx1)
153 end
154 --- *****
155 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-6, iterations=10000})
156 transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.2, 0.36, 0.7)
157 transientanalysis:addcallback(displpertime, "timestep")
158 transientanalysis:addcallback(reactpertime, "timestep")
159 model:setRayleighCoefficients(0.6318799279194399, 0.00015503501814608252)
160 transientanalysis:solve(7641)
161 --- *****
162 function writedata1(x, fname)
163     local f = assert(io.open(fname, 'w'))
164     local writenl = 0
165     for i,v in ipairs(x) do
166         f:write(v, " ")
167         writenl = writenl + 1
168         -- length of row size: writenl
169         if (writenl > 0) then
170             writenl = 0
171             f:write("\n")
172         end
173     end
174     f:close()
175 end
176 writedata1(displ, 'Ex27HHTNodalDisp_5.dat')
177 writedata1(react, 'Ex27HHTReact_1.dat')
178 print("Transient analysis ended\n")

```

9.3 MATLAB

```

1 %=====  

2 % Mercury Matlab Version 1.0.1  

3 % Written by Dae-Hung Kang, CU-NEES  

4 % Copyright 2009, CU-NEES  

5 % Written : October 2009.  

6 % File name: Ex28.m (1 column)  

7 %=====  

8 % Preface  

9 Unit = {'kip', 'in'};  

10 % ndim, ndofpn  

11 StrMode = {2, 3};  

12 %=====  

13 % Control block  

14 Iteration = {'static', { {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};  

15     {'InitialStiffness', 10000, 1.0e-6, 'DisplNorm'};  

16     }  

17     'element', { {'NewtonRaphson', 10, 1.0e-6, 'DisplNorm'};  

18     {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};  

19     }  

20     'transient', { {'NewtonRaphson', 10, 1.0e-6, 'DisplNorm'};  

21     {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};  

22     }  

23     };  

24 Integration = {'HHT', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252};  

25 addMass = {1, 0, 0, 0;  

26     2, 2.50712E-05, 2.50712E-05, 0;

```

```

27     3, 0.000137891, 0.000137891, 0;
28     4, 0.000137891, 0.000137891, 0;
29     5, 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0};
30 %
31 nodcoord = {1, 0, 0;
32             2, 0, 0;
33             3, 0, 6;
34             4, 0, 33;
35             5, 0, 39};
36 constraint = {1, 1, 1, 1};
37 %
38
39 elements = { {1, 'ZeroLength2D', 1, 2, 0, 26, 0, pi()/2};
40             {2, 'ZeroLength2DSection', 1, 2, pi()/2, 3};
41             {3, 'StiffnessBased2DBeamColumn', 2, 3, 3, 1};
42             {4, 'FlexibilityBased2DBeamColumn1', 3, 4, 5, 1};
43             {5, 'StiffnessBased2DBeamColumn', 4, 5, 3, 1}};
44 %
45 sections = {1, 'Layer', {1, 5.1, 2.575;
46                       1, 5.1, -2.575;
47                       1, 0.91375, 1.88125;
48                       1, 0.91375, 1.34375;
49                       1, 0.91375, 0.80625;
50                       1, 0.91375, 0.26875;
51                       1, 0.91375, -0.26875;
52                       1, 0.91375, -0.80625;
53                       1, 0.91375, -1.34375;
54                       1, 0.91375, -1.88125;
55                       2, 2.31125, 1.88125;
56                       2, 2.31125, 1.34375;
57                       2, 2.31125, 0.80625;
58                       2, 2.31125, 0.26875;
59                       2, 2.31125, -0.26875;
60                       2, 2.31125, -0.80625;
61                       2, 2.31125, -1.34375;
62                       2, 2.31125, -1.88125;
63                       17, 0.147, 2.15;
64                       17, 0.098, 0;
65                       17, 0.147, -2.15};
66     3, 'Layer', {5, 5.1, 2.575;
67                5, 5.1, -2.575;
68                5, 0.91375, 1.88125;
69                5, 0.91375, 1.34375;
70                5, 0.91375, 0.80625;
71                5, 0.91375, 0.26875;
72                5, 0.91375, -0.26875;
73                5, 0.91375, -0.80625;
74                5, 0.91375, -1.34375;
75                5, 0.91375, -1.88125;
76                6, 2.31125, 1.88125;
77                6, 2.31125, 1.34375;
78                6, 2.31125, 0.80625;
79                6, 2.31125, 0.26875;
80                6, 2.31125, -0.26875;
81                6, 2.31125, -0.80625;
82                6, 2.31125, -1.34375;
83                6, 2.31125, -1.88125;
84                19, 0.147, 2.15;
85                19, 0.098, 0;
86                19, 0.147, -2.15}}};
87 %
88 materials = { {1, 'AnisotropicDamage', 2829, 0.2, 5.855E-08, 1870, 0.003248, 0.05168, 38890000000, 22.27, 0.9999999, 0};
89             { 2, 'AnisotropicDamage', 1804, 0.2, 0.00006483, 503.6, 3.125E-07, 0.4054, 3159000000, 102, 0.9999999, 0};
90             { 5, 'AnisotropicDamage', 569.3, 0.2, 1.689E-19, 371.8, 0.832, 1.461, 9.529E+13, 100.4, 0.9999999, 0};
91             { 6, 'AnisotropicDamage', 396.7, 0.2, 0.0008326, 124.8, 2.084E-07, 1.61, 8.856E+11, 62.48, 0.9999999, 0};
92             { 17, 'ModGMP', 26500, 87.5, 0.01, 15, 0.925, 0.15, 0, 0, 55, 0, 55};
93             { 19, 'ModGMP', 6949, 87.5, 0.01, 15, 0.925, 0.15, 0, 0, 55, 0, 55};
94             { 26, 'Bilinear', 1690, 78.2, 0.173, 0, 0, 55, 0, 55}}};
95 %
96 % Force block
97 ga = load('NR_g-dt-0.01-Matlab.txt');
98 nga = size(ga, 1);
99 for i = 1:nga
100     groundacceleration{i,1} = ga(i,1);
101     groundacceleration{i,2} = ga(i,2);
102     groundacceleration{i,3} = ga(i,3);
103 end
104 forces = { 1, 'Static', {'NodalForces', {5, 1, 0}};
105           2, 'Acceleration', {386.4, groundacceleration}};

```

9.4 C++

```

1  --- *****
2  --- 1 column
3  --- *****
4  --- o (39)
5  --- | Stiffness-based beam-column with 2 integration points
6  --- o (33)
7  --- |
8  --- | Flexibility-based beam-column with 5 integration points
9  --- |
10 --- o (6)
11 --- | Stiffness-based beam-column with 2 integration points
12 --- o (0)
13 --- | Zero-Length and zero-Length section elements (bottom bar slip and shear deformation)
14 --- .o. (0) (Fixed support)
15
16 --- *****
17 elements = {}
18
19 --- create ductile column node coordinates
20 --- create nodes
21 nodes = {{1, 0, 0, 'mass', 0, 0, 0}};

```

```

22 {2, 0, 0, 'mass', 2.50712E-05, 2.50712E-05, 0};
23 {3, 0, 6, 'mass', 0.000137891, 0.000137891, 0};
24 {4, 0, 33, 'mass', 0.000137891, 0.000137891, 0};
25 {5, 0, 39, 'mass', 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0 }
26 --- figure out section names
27 barslipsectionf = 'BSColDFSection'
28 barslipspringf = 'BSColDFSS'
29 columnsection = 'ColDSection'
30 columnrigidsection = 'ColRigidSection'
31 nIp_stif = 3;
32 nIp_flex = 5;
33 flexparams = {10000, 1e-3}
34 --- Define elements
35 barslipbottom = { 1, 'InterfaceElement2D', 1, 2, { {barslipspringf, {1,0,0} } },
36 { {barslipsectionf}}, {0,1,0},{-1,0,0} }
37 plasticcolumn1 = { 2, 'StiffnessBased2DBeamColumn', 2, 3, {columnsection, nIp_stif} }
38 flexcolumn = { 3, 'FlexibilityBased2DBeamColumn', 3, 4, {columnsection, nIp_flex}, flexparams }
39 plasticcolumn2 = { 4, 'StiffnessBased2DBeamColumn', 4, 5, {columnsection, nIp_stif} }
40 table.insert(elements, barslipbottom)
41 table.insert(elements, plasticcolumn1)
42 table.insert(elements, flexcolumn)
43 table.insert(elements, plasticcolumn2)
44 --- *****
45 --- Set section properties.
46 ---
47 allsections = {
48 'ColDSection', 'Fiber',
49 --- Tag, Area, y-loc, z-loc
50 {
51 'ColDCover', 5.1, 2.575, 0,
52 'ColDCover', 5.1, -2.575, 0,
53 'ColDCover', 0.91375, 1.88125, 0,
54 'ColDCover', 0.91375, 1.34375, 0,
55 'ColDCover', 0.91375, 0.80625, 0,
56 'ColDCover', 0.91375, 0.26875, 0,
57 'ColDCover', 0.91375, -0.26875, 0,
58 'ColDCover', 0.91375, -0.80625, 0,
59 'ColDCover', 0.91375, -1.34375, 0,
60 'ColDCover', 0.91375, -1.88125, 0,
61 'ColDCore', 2.31125, 1.88125, 0,
62 'ColDCore', 2.31125, 1.34375, 0,
63 'ColDCore', 2.31125, 0.80625, 0,
64 'ColDCore', 2.31125, 0.26875, 0,
65 'ColDCore', 2.31125, -0.26875, 0,
66 'ColDCore', 2.31125, -0.80625, 0,
67 'ColDCore', 2.31125, -1.34375, 0,
68 'ColDCore', 2.31125, -1.88125, 0,
69 'ColDSteel', 0.147, 2.15, 0,
70 'ColDSteel', 0.098, 0, 0,
71 'ColDSteel', 0.147, -2.15, 0 };
72 ---
73 'BSColDFSection', 'Fiber',
74 {
75 'BSColDFCover', 5.1, 2.575, 0,
76 'BSColDFCover', 5.1, -2.575, 0,
77 'BSColDFCover', 0.91375, 1.88125, 0,
78 'BSColDFCover', 0.91375, 1.34375, 0,
79 'BSColDFCover', 0.91375, 0.80625, 0,
80 'BSColDFCover', 0.91375, 0.26875, 0,
81 'BSColDFCover', 0.91375, -0.26875, 0,
82 'BSColDFCover', 0.91375, -0.80625, 0,
83 'BSColDFCover', 0.91375, -1.34375, 0,
84 'BSColDFCover', 0.91375, -1.88125, 0,
85 'BSColDFCore', 2.31125, 1.88125, 0,
86 'BSColDFCore', 2.31125, 1.34375, 0,
87 'BSColDFCore', 2.31125, 0.80625, 0,
88 'BSColDFCore', 2.31125, 0.26875, 0,
89 'BSColDFCore', 2.31125, -0.26875, 0,
90 'BSColDFCore', 2.31125, -0.80625, 0,
91 'BSColDFCore', 2.31125, -1.34375, 0,
92 'BSColDFCore', 2.31125, -1.88125, 0,
93 'BSColDFSteel', 0.147, 2.15, 0,
94 'BSColDFSteel', 0.098, 0, 0,
95 'BSColDFSteel', 0.147, -2.15, 0 }; }
96 --- *****
97 --- Set material properties.
98 concretemat = 'ConcreteLinearTensionSoftening'
99 steelmat = 'GiuffreMenegottoPinto'
100 sspringmat = 'Bilinear'
101 materials = {
102 {'ColDCover', 'anisotropicdamage2', 2829, 0, 0.2, 5.855E-08, 1870, 0.003248, 0.05168, 3889000000, 22.27, 0.99999999};
103 {'ColDCore', 'anisotropicdamage2', 1804, 0, 0.2, 0.000006483, 503.6, 3.125E-07, 0.4054, 3159000000, 102, 0.99999999};
104 {'BSColDFCover', 'anisotropicdamage2', 569.3, 0, 0.2, 1.689E-19, 371.8, 0.832, 1.461, 9.529E+13, 100.4, 0.99999999};
105 {'BSColDFCore', 'anisotropicdamage2', 396.7, 0, 0.2, 0.0008326, 124.8, 2.084E-07, 1.61, 8.856E+11, 62.48, 0.99999999 };
106 {'ColDSteel', steelmat, 26500, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
107 {'BSColDFSteel', steelmat, 6949, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
108 {'BSColDFSS', sspringmat, 1690, 0, 78.2, 0.173, 0, 55, 0, 55},
109 }
110 --- *****
111 function dumptable(x)
112 local result = '{'
113 for i,v in ipairs(x) do
114 if (type(v) == 'table') then
115 result = result .. dumptable(v)
116 else
117 result = result .. v .. ','
118 end
119 end
120 return result .. '}'\n'
121 end
122 ---print(dumptable(nodes))
123 ---print(dumptable(elements))
124 ---print(dumptable(materials))
125 ---print(dumptable(allsections))
126 --- *****
127 --- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
128 model = StructureModel(2,3)

```



```

129 --- Assign all input data to Mercury
130 model:addNodes(nodes)
131 model:addMaterials(materials)
132 model:addSections(allsections)
133 model:addElements(elements)
134 --- *****
135 --- constrain bottom nodes
136 model:constrainNode(1,1,1,1)
137 --- *****
138 print(" Transient analysis started\n")
139 earthquakeloading = LoadDescription()
140 earthquakeloading:addLoad({'groundmotion', 'NR_g_dt_0_01_OpneSees.txt', dt=0.01', 1, 386.4})
141 --- *****
142 displ = {}
143 function displertime(time)
144     dx5,dy5,dz5 = model:nodeDisplacements(5)
145     table.insert(displ, dx5)
146     print("Work\n");
147 end
148 ---
149 react = {}
150 function reactptime(time)
151     fx1,fy1 = model:nodeRestoringForces(1)
152     table.insert(react, fx1)
153 end
154 --- *****
155 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=100000})
156 transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.2, 0.36, 0.7)
157 transientanalysis:addcallback(displertime, "timestep")
158 transientanalysis:addcallback(reactptime, "timestep")
159 model:setRayleighCoefficients(0.6318799279194399, 0.00015503501814608252)
160 transientanalysis:solve(7641)
161 --- *****
162 function writedata1(x, fname)
163     local f = assert(io.open(fname, 'w'))
164     local writenl = 0
165     for i,v in ipairs(x) do
166         f:write(v, " ")
167         writenl = writenl + 1
168         --- length of row size: writenl
169         if (writenl > 0) then
170             writenl = 0
171             f:write("\n")
172         end
173     end
174     f:close()
175 end
176 writedata1(displ, 'Ex28HHTNodalDisp_5.dat')
177 writedata1(react, 'Ex28HHTReact_1.dat')
178 print(" Transient analysis ended\n")

```

10 Full Reinforced Concrete Frame, HHT, Shing

This most complex validation example, Fig. 12, analyses the reinforced concrete frame tested on a shake table by ?, Fig. 10, and which will be later used to assess the real time hybrid simulation of Mercury, Fig. 13.

This example is sufficiently complex to warrant detailed description.

10.0.1 Frame discretization and properties

Beams and columns are modeled with one stiffness-based beam-columns (3 integration points) at each end, and one flexibility-based beam-column with 5 integrations.

Nodes 7, 14, 21 and 49 are monitored, in Fig. 12 to assess the seismic while ignoring self-weight. Columns **A** and **B** are shear critical (i.e under-reinforced), while **C** and **D** are ductile. Sectional characteristics are shown in Fig. 15 while Table 6 to 9 summarize those properties and constitutive model parameters used¹.

Nodal masses are used in the dynamic analysis, Table 10 to 12, where each beam has an added lead bundle masses at 4th and 5th nodes. For comparison HHT integration scheme is used in OpenSees, and HHT and Shing integration scheme are used in Mercury c++ version with $\alpha = -0.2$. This last model is particularly relevant for the real-time hybrid simulation which will be conducted later. Rayleigh damping coefficients were determined to be 1.177 for mass and 0.001599 for stiffness. The reinforced concrete frame is subjected to the seismic excitation shown in Fig. 16 (which was measured at the base of the shake table in Ghannoum's tests).

10.1 MATLAB

1 xxx

¹Col: Column, Beam: Beam, F: Footing section, D: Ductile section, ND: Shear critical section, BS: Bar-Slip section, Rigid: Rigid section, Cover: Concrete cover fiber, Core: Concrete core fiber, SS: Shear spring of zero-length element, steel: Reinforcement fiber, ModKP: modified Kent-Park model, ModGMP: modified Giuffre-Monegotto-Pinto

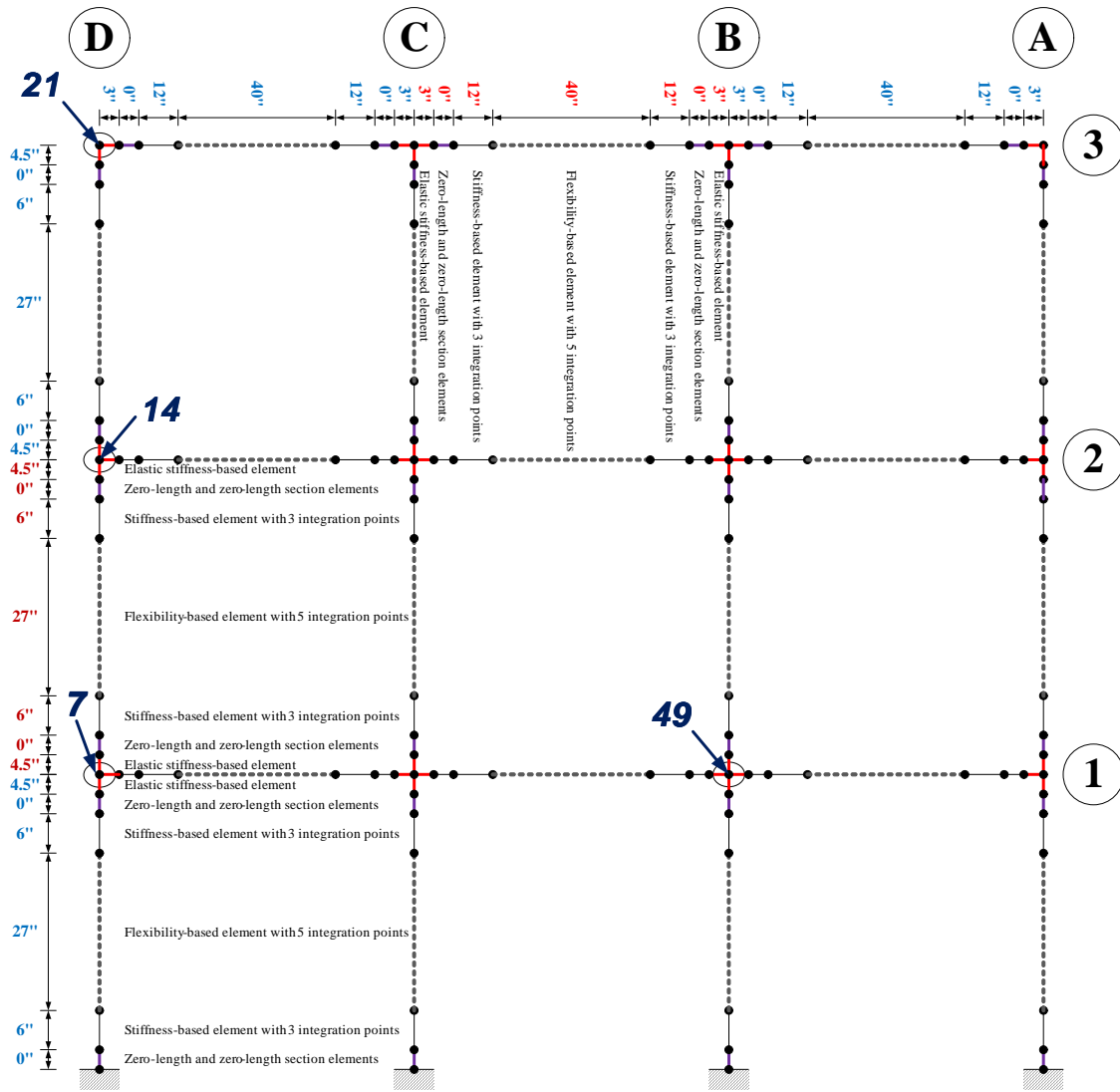


Figure 12: Numerical model for real-time hybrid simulation

Table 6: Concrete material properties of concrete for Ex29 (unit:kips, in)

Fiber name	Material	$E_t s$	σ_c	ϵ_c	σ_{cu}	ϵ_{cu}	λ	σ_t
Col-D-Cover	ModKP	549.23	-3.57	-0.0026	-1.19	-0.0078	0.3	0.4481
Col-D-Core	ModKP	549.45	-7.5	-0.00546	-7.35	-0.01638	0.3	0.6495
BS-Col-D-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Col-D-Core	ModKP	104.17	-7.5	-0.0288	-7.35	-0.0864	0.3	0.6495
BS-Col-D-F-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Col-D-F-Core	ModKP	104.17	-7.5	-0.0288	-6.75	-0.0864	0.3	0.6495
Col-ND-Cover	ModKP	549.23	-3.57	-0.0026	-1.19	-0.0078	0.3	0.4481
Col-ND-Core	ModKP	549.30	-3.9	-0.00284	-3.51	-0.00852	0.3	0.4684
BS-Col-ND-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Col-ND-Core	ModKP	106.85	-3.9	-0.0146	-3.51	-0.0438	0.3	0.4684
BS-Col-ND-F-Cover	ModKP	144.24	-3.57	-0.0099	-1.19	-0.0297	0.3	0.4481
BS-Col-ND-F-Core	ModKP	106.85	-3.9	-0.0146	-3.51	-0.0438	0.3	0.4684
Beam-Cover	ModKP	549.23	-3.57	-0.0026	-1.19	-0.0078	0.3	0.4481
Beam-Core	ModKP	549.30	-3.9	-0.00284	-3.51	-0.00852	0.3	0.4684
BS-Beam-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Beam-Core	ModKP	106.85	-3.9	-0.0146	-3.51	-0.0438	0.3	0.4684

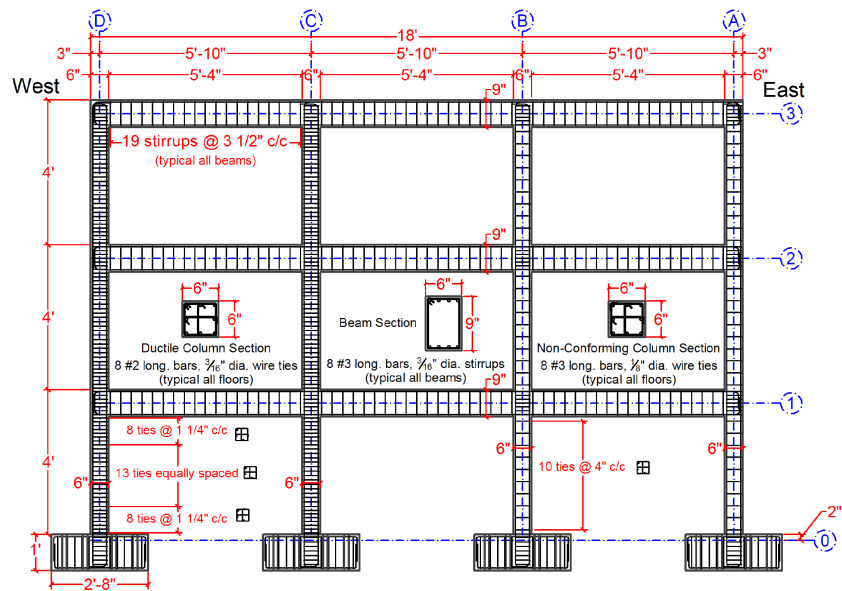


Figure 13: Reinforced concrete details, ?

Figure 14: Shake table test of reinforce concrete frame, ?

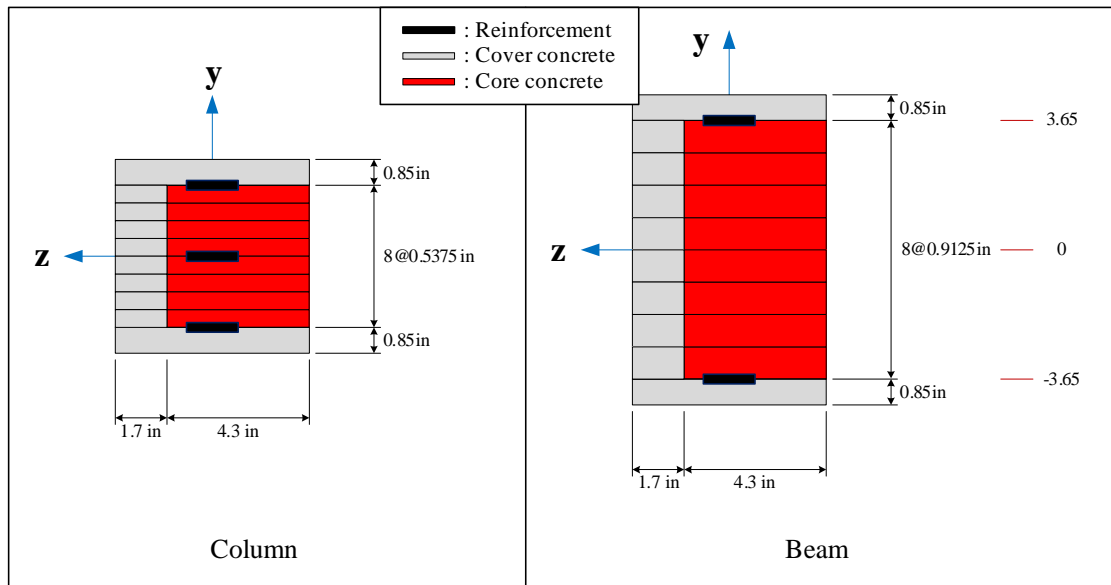


Figure 15: Section properties of Ex29

Table 7: Material properties of reinforcement for Ex29 (unit: kips, in)

Fiber name	Material	E	σ_y	b	$R0$	$cR1$	$cR2$	$a1$	$a2$	$a3$	$a4$
Col-D-Steel	ModGMP	26500	87.5	0.01	15	0.925	0.15	0	55	0	55
BS-Col-D-Steel	ModGMP	5067	87.5	0.01	15	0.925	0.15	0	55	0	55
BS-Col-D-F-Steel	ModGMP	6949	87.5	0.01	15	0.925	0.15	0	55	0	55
Col-ND-Steel	ModGMP	27300	80	0.01	15	0.925	0.15	0	55	0	55
BS-Col-ND-Steel	ModGMP	5220	80	0.01	15	0.925	0.15	0	55	0	55
BS-Col-ND-F-Steel	ModGMP	5220	80	0.01	15	0.925	0.15	0	55	0	55
Beam-Steel	ModGMP	27300	80	0.01	15	0.925	0.15	0	55	0	55
BS-Beam-Steel	ModGMP	5220	80	0.01	15	0.925	0.15	0	55	0	55

Table 8: Property of shear spring for Ex29 (unit: kips, in)

Fiber name	Material	E	σ_y	b	$a1$	$a2$	$a3$	$a4$
BS-Col-D-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Col-D-F-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Col-ND-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Col-ND-F-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Beam-SS	Bilinear	1545	75.8	0.396	0	55	0	55

Table 9: Properties of rigid elements (unit: kips, in)

Element	Model	E
Col-Rigid	Elastic	1E+10
Beam-Rigid	Elastic	1E+10

Figure 16: Seismic excitation for Ex29

Table 10: Mass properties in first floor column nodes (unit: $kips \cdot sec^2/in$)

First floor node	m_x	m_y	m_z
7th	0.0000752	0.0000752	0.00000752
6th	0.0000188	0.0000188	0.00000188
5th	0.0000251	0.0000251	0.00000251
4th	0.0001379	0.0001379	0.00001379
3rd	0.0001379	0.0001379	0.00001379
2nd	0.0000251	0.0000251	0.00000251
1st	0.0000000	0.0000000	0.00000000

Table 11: Mass properties in column nodes except first floor columns (unit: $kips \cdot sec^2/in$)

Other floor node	m_x	m_y	m_z
8th	0.0000627	0.0000627	0.00000627
7th	0.0000188	0.0000188	0.00000188
6th	0.0000251	0.0000251	0.00000251
5th	0.0001379	0.0001379	0.00001379
4th	0.0001379	0.0001379	0.00001379
3rd	0.0000251	0.0000251	0.00000251
2nd	0.0000188	0.0000188	0.00000188

Table 12: Mass properties in beam nodes (unit: $kips \cdot sec^2/in$)

Beam node	m_x	m_y	m_z
2nd	0.0000188	0.0000188	0.00000188
3rd	0.0000752	0.0000752	0.00000752
4th	0.0080899	0.0080899	0.00080899
5th	0.0080899	0.0080899	0.00080899
6th	0.0000752	0.0000752	0.00000752
7th	0.0000188	0.0000188	0.00000188

10.2 C++

```

1 *****
2 earthquakefiles = {
3     'Test16AccelHHalfYield.txt',
4     'Test19AccelH1stCollapse.txt',
5     'Test30AccelH2ndCollapse.txt',
6     'Test32AccelH3rdCollapse.txt'
7 }
8 earthquakefile = earthquakefiles[2]
9 print('Excitation:', earthquakefile)
10 -----
11 alpha = -0.2
12 beta = (1-alpha)*(1-alpha)/4
13 gamma = (1-2*alpha)/2
14 -- alpha : alpha coefficient in HHT integration
15 -- beta : beta coefficient in HHT integration
16 -- gamma : gamma coefficient in HHT integration
17 -----
18 -- Multiple bay/floor building model
19 -----
20 -- Column : first floor
21 -- o (43.5)
22 -- | Elastic stiffness-based beam-column
23 -- o (39)
24 -- | Zero-Length and zero-Length section elements (top bar slip and shear deformation)
25 -- o (39)
26 -- | Stiffness-based beam-column with 2 integration points
27 -- o (33)
28 -- |
29 -- | Flexibility-based beam-column with 5 integration points
30 -- |
31 -- o (6)
32 -- | Stiffness-based beam-column with 2 integration points
33 -- o (0)
34 -- | Zero-Length and zero-Length section elements (bottom bar slip and shear deformation)
35 -- o- (0) (Fixed support)
36 -----
37 -----
38 -- Column : other floor
39 -- o (48)
40 -- | Elastic stiffness-based beam-column
41 -- o (43.5)
42 -- | Zero-Length and zero-Length section elements (top bar slip and shear deformation)
43 -- o (43.5)
44 -- | Stiffness-based beam-column with 2 integration points
45 -- o (37.5)
46 -- |
47 -- | Flexibility-based beam-column with 5 integration points
48 -- |
49 -- o (10.5)
50 -- | Stiffness-based beam-column with 2 integration points
51 -- o (4.5)
52 -- | Zero-Length and zero-Length section elements (bottom bar slip and shear deformation)
53 -- o (4.5)
54 -- | Elastic stiffness-based beam-column
55 -- o (0)
56 -----
57 -- Beam
58 -- o-----o-----o-----o-----o-----o-----o-----o
59 -- (0) a (3) b (3) c (15) d (55)e(67) f (67) g (70)
60 -- a: Elastic stiffness-based beam-column
61 -- b: Zero-Length and zero-Length section elements (left bar slip and shear deformation)
62 -- c: Stiffness-based beam-column with 2 integration points
63 -- d: Flexibility-based beam-column with 5 integration points
64 -- e: Stiffness-based beam-column with 2 integration points
65 -- f: Zero-Length and zero-Length section elements (right bar slip and shear deformation)
66 -- g: Elastic stiffness-based beam-column
67 -----
68 -- Basic variables for node information
69 -- g : gravity acceleration
70 -- nbays : Number of bays (nbays must be odd number)
71 -- nflrs : Number of floors
72 -- nlcnod: Number of nodes of the first column
73 -- nocnod: Number of nodes of other columns
74 -- nbmnod: Number of nodes of beams
75 -- clen : Total length of the first column
76 -- colen : Total length of other columns
77 -- bmlen : Total length of beams
78 -- clnod : Coordinate information on the first column
79 -- conod : Coordinate information on other columns
80 -- bmnod : Coordinate information on beams
81 -- cdlnodmass : Nodal mass of the first ductile column
82 -- cdonodmass : Nodal mass of other ductile columns
83 -- cndlnodmass : Nodal mass of the first shear critical column
84 -- cndonodmass : Nodal mass of other shear critical column
85 -- nlp_elas : Number of integration points of elastic stiffness-based beam-column
86 -- nlp_stif : Number of integration points of nonlinear stiffness-based beam-column
87 -- nlp_flex : Number of integration points of nonlinear flexibility-based beam-column3
88 -- flexparams : Iteration config for flexibility elements
89 -- flexparams = {number of iterations, tolerance}
90 -- $$$ start of node and element information $$$
91 g = 386.4
92 nbays = 3; -- nbays must be odd number.
93 nflrs = 3;
94 nlcnod = 7;
95 nocnod = 8;
96 nbmnod = 8;
97 clen = 43.5;
98 colen = 48;
99 bmlen = 70;
100 clnod = {0, 0, 6, 33, 39, 39, 43.5};
101 conod = {0, 4.5, 4.5, 10.5, 37.5, 43.5, 43.5, 48};
102 bmnod = {0, 3, 3, 15, 55, 67, 67, 70};
103 bmaddWeight = 3
104 bmaddMass = bmaddWeight/g

```

```

105 -----
106 c1nodmass = {0.0, 2.50712E-05, 0.000137891, 0.000137891, 2.50712E-05, 1.88034E-05, 7.52135E-05}
107 conodmass = {0.0, 1.88034E-05, 2.50712E-05, 0.000137891, 0.000137891, 2.50712E-05, 1.88034E-05, 6.26779E-05}
108 bmnodmass = {0.0, 1.88034E-05, 7.52135E-05, 0.000325925+bmaddMass, 0.000325925+bmaddMass,
109 7.52135E-05, 1.88034E-05, 0.0}
110 -----
111 nlp_elas = 1;
112 nlp_stif = 3;
113 nlp_flex = 5;
114 -----
115 flexparams = {1000,1e-6}
116 --- $$$ END of node and element information $$$
117 -----
118 --- Set nodes and elements.
119 -----
120 -----
121 nodes = {}
122 elements = {}
123 -----
124 function nodename(bay, floor)
125     return string.format('node-%d-%d', bay, floor)
126 end
127 -----
128 function columnnodename(bay, floor, subsection)
129     if (subsection == 1 and floor > 1) then
130         return columnnodename(bay, floor -1, nocnod)
131     if (floor == 2) then
132         return columnnodename(bay, floor -1, n1cnod)
133     else
134         return columnnodename(bay, floor -1, nocnod)
135     end
136     else
137         return string.format('colnode-%d-%d-%d', bay, floor, subsection)
138     end
139 end
140 -----
141 function beamnodename(bay, floor, subsection)
142     --- section 1 nodes re-use the column nodes
143     if (subsection == 1) then
144         return columnnodename(bay, floor, 1)
145     else
146         return string.format('beamnode-%d-%d-%d', bay, floor, subsection)
147     end
148 end
149 -----
150 -----
151 --- create ductile column node coordinates
152 ncold = (nbays+1)/2;
153 for colbay = 1,ncold do
154     for colfloor = 1,nflrs do
155         x = (colbay-1)*bmlen
156         nodeyfirst = c1nod
157         nodeyhigher = conod
158         --- create nodes
159         if (colfloor==1) then
160             bottomnode = columnnodename(colbay, colfloor, 1)
161             for k=1,n1cnod do
162                 y = nodeyfirst[k]
163                 masscd1 = c1nodmass[k]
164                 curnode = { columnnodename(colbay, colfloor, k), x,y, 'mass', masscd1, masscd1, 0.1*masscd1 }
165                 table.insert(nodes, curnode)
166             end
167         else
168             bottomnode = columnnodename(colbay, colfloor -1, nocnod)
169             for k=2,nocnod do
170                 y = c1len + colen * (colfloor -2) + nodeyhigher[k]
171                 masscd0 = conodmass[k]
172                 curnode = { columnnodename(colbay, colfloor, k), x,y, 'mass', masscd0, masscd0, 0.1*masscd0 }
173                 table.insert(nodes, curnode)
174             end
175         end
176         --- figure out section names
177         barslipsectionf = 'BSColDFSection'
178         barslipspringf = 'BSColDFSS'
179         columnsection = 'ColDSection'
180         barslipspring = 'BSColDSS'
181         barslipsection = 'BSColDSection'
182         columnrigidsection = 'ColRigidSection'
183         if (colfloor==1) then
184             node1 = columnnodename(colbay, colfloor, 1)
185             node2 = columnnodename(colbay, colfloor, 2)
186             node3 = columnnodename(colbay, colfloor, 3)
187             node4 = columnnodename(colbay, colfloor, 4)
188             node5 = columnnodename(colbay, colfloor, 5)
189             node6 = columnnodename(colbay, colfloor, 6)
190             node7 = columnnodename(colbay, colfloor, 7)
191             --- Define elements
192             barslipbottom = { string.format('columnbsb-%d-%d', colbay, colfloor), 'InterfaceElement2D', node1, node2,
193                 { {barslipspringf, {1,0,0}} }, { {barslipsectionf}, {0,1,0}, {-1,0,0} } }
194             plasticcolumn1 = { string.format('columnpl1-%d-%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
195                 node2, node3, {columnsection, nlp_stif} }
196             flexcolumn = { string.format('columnflx-%d-%d', colbay, colfloor), 'FlexibilityBased2DBeamColumn',
197                 node3, node4, {columnsection, nlp_flex}, flexparams }
198             plasticcolumn2 = { string.format('columnpl2-%d-%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
199                 node4, node5, {columnsection, nlp_stif} }
200             barsliptop = { string.format('columnbst-%d-%d', colbay, colfloor), 'InterfaceElement2D', node5, node6,
201                 { {barslipspring, {1,0,0}} }, { {barslipsection}, {0,1,0}, {-1,0,0} } }
202             rigidcolumn2top = { string.format('columnrdt-%d-%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
203                 node6, node7, {columnrigidsection, nlp_elas} }
204             table.insert(elements, barslipbottom)
205             table.insert(elements, plasticcolumn1)
206             table.insert(elements, flexcolumn)
207             table.insert(elements, plasticcolumn2)
208             table.insert(elements, barsliptop)
209             table.insert(elements, rigidcolumn2top)
210         else
211             if (colfloor == 2) then

```

```

212     node1 = columnnodename(colbay, colfloor-1, nlnod)
213     else
214     node1 = columnnodename(colbay, colfloor-1, nocnod)
215     end
216     node2 = columnnodename(colbay, colfloor, 2)
217     node3 = columnnodename(colbay, colfloor, 3)
218     node4 = columnnodename(colbay, colfloor, 4)
219     node5 = columnnodename(colbay, colfloor, 5)
220     node6 = columnnodename(colbay, colfloor, 6)
221     node7 = columnnodename(colbay, colfloor, 7)
222     node8 = columnnodename(colbay, colfloor, 8)
223     -- Define elements
224     rigidcolumnbottom = { string.format('columnrdb_%d_%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
225     node1, node2, {columnrigidsection, nlp_elas} }
226     barslipbottom = { string.format('columnbsb_%d_%d', colbay, colfloor), 'InterfaceElement2D', node2,
227     node3, { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
228     plasticcolumn1 = { string.format('columnpl1_%d_%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
229     node3, node4, {columnsection, nlp_stif} }
230     flexcolumn = { string.format('columnflx_%d_%d', colbay, colfloor), 'FlexibilityBased2DBeamColumn',
231     node4, node5, {columnsection, nlp_flex}, flexparams }
232     plasticcolumn2 = { string.format('columnpl2_%d_%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
233     node5, node6, {columnsection, nlp_stif} }
234     barsliptop = { string.format('columnbst_%d_%d', colbay, colfloor), 'InterfaceElement2D', node6,
235     node7, { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
236     rigidcolumnmtop = { string.format('columnrdt_%d_%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
237     node7, node8, {columnrigidsection, nlp_elas} }
238     table.insert(elements, rigidcolumnbottom)
239     table.insert(elements, barslipbottom)
240     table.insert(elements, plasticcolumn1)
241     table.insert(elements, flexcolumn)
242     table.insert(elements, plasticcolumn2)
243     table.insert(elements, barsliptop)
244     table.insert(elements, rigidcolumnmtop)
245     end
246     end
247 end
248
249 -- create shear critical column node coordinates
250 for colbay = ncold+1, nbays+1 do
251     for colfloor = 1, nflrs do
252         x = (colbay-1)*bmlen
253         nodeyfirst = clnod
254         nodeyhigher = conod
255         -- create nodes
256         if (colfloor==1) then
257             bottomnode = columnnodename(colbay, colfloor, 1)
258             for k=1, nlnod do
259                 y = nodeyfirst[k]
260                 masscnd1 = clnodmass[k]
261                 curnode = { columnnodename(colbay, colfloor, k), x, y, 'mass', masscnd1, masscnd1, 0.1*masscnd1 }
262                 table.insert(nodes, curnode)
263             end
264         else
265             bottomnode = columnnodename(colbay, colfloor-1, nocnod)
266             for k=2, nocnod do
267                 y = cilen + colen * (colfloor-2) + nodeyhigher[k]
268                 masscndo = conodmass[k]
269                 curnode = { columnnodename(colbay, colfloor, k), x, y, 'mass', masscndo, masscndo, 0.1*masscndo }
270                 table.insert(nodes, curnode)
271             end
272         end
273         -- figure out section names
274         barslipsectionf = 'BSColNDFSection'
275         barslipspringf = 'BSColNDFSS'
276         columnsection = 'ColNDFSection'
277         barslipspring = 'BSColNDSS'
278         barslipsection = 'BSColNDFSection'
279         columnrigidsection = 'ColRigidSection'
280         if (colfloor==1) then
281             node1 = columnnodename(colbay, colfloor, 1)
282             node2 = columnnodename(colbay, colfloor, 2)
283             node3 = columnnodename(colbay, colfloor, 3)
284             node4 = columnnodename(colbay, colfloor, 4)
285             node5 = columnnodename(colbay, colfloor, 5)
286             node6 = columnnodename(colbay, colfloor, 6)
287             node7 = columnnodename(colbay, colfloor, 7)
288             -- Define elements
289             barslipbottom = { string.format('columnbsb_%d_%d', colbay, colfloor), 'InterfaceElement2D',
290             node1, node2, { {barslipspringf, {1,0,0} } }, { {barslipsectionf}}, {0,1,0},{-1,0,0} }
291             plasticcolumn1 = { string.format('columnpl1_%d_%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
292             node2, node3, {columnsection, nlp_stif} }
293             flexcolumn = { string.format('columnflx_%d_%d', colbay, colfloor), 'FlexibilityBased2DBeamColumn',
294             node3, node4, {columnsection, nlp_flex}, flexparams }
295             plasticcolumn2 = { string.format('columnpl2_%d_%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
296             node4, node5, {columnsection, nlp_stif} }
297             barsliptop = { string.format('columnbst_%d_%d', colbay, colfloor), 'InterfaceElement2D', node5, node6,
298             { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
299             rigidcolumnmtop = { string.format('columnrdt_%d_%d', colbay, colfloor), 'StiffnessBased2DBeamColumn',
300             node6, node7, {columnrigidsection, nlp_elas} }
301             table.insert(elements, barslipbottom)
302             table.insert(elements, plasticcolumn1)
303             table.insert(elements, flexcolumn)
304             table.insert(elements, plasticcolumn2)
305             table.insert(elements, barsliptop)
306             table.insert(elements, rigidcolumnmtop)
307         else
308             if (colfloor == 2) then
309                 node1 = columnnodename(colbay, colfloor-1, nlnod)
310             else
311                 node1 = columnnodename(colbay, colfloor-1, nocnod)
312             end
313             node2 = columnnodename(colbay, colfloor, 2)
314             node3 = columnnodename(colbay, colfloor, 3)
315             node4 = columnnodename(colbay, colfloor, 4)
316             node5 = columnnodename(colbay, colfloor, 5)
317             node6 = columnnodename(colbay, colfloor, 6)
318             node7 = columnnodename(colbay, colfloor, 7)

```

```

319     node8 = columnnodename(colbay , colfloor ,8)
320     -- Define elements
321     rigidcolumnbottom = { string.format('columnrdb_%d_%d',colbay , colfloor), 'StiffnessBased2DBeamColumn',
322         node1, node2, {columnrigidsection , nIp_elas} }
323     barslipbottom = { string.format('columnbsb_%d_%d',colbay , colfloor), 'InterfaceElement2D',
324         node2, node3, { {barslipspring , {1,0,0} } }, { {barslipsection} }, {0,1,0},{-1,0,0} }
325     plasticcolumn1 = { string.format('columnpl1_%d_%d',colbay , colfloor), 'StiffnessBased2DBeamColumn',
326         node3, node4, {columnsection , nIp_stif} }
327     flexcolumn = { string.format('columnflx_%d_%d',colbay , colfloor), 'FlexibilityBased2DBeamColumn',
328         node4, node5, {columnsection , nIp_flex}, flexparams }
329     plasticcolumn2 = { string.format('columnpl2_%d_%d',colbay , colfloor), 'StiffnessBased2DBeamColumn',
330         node5, node6, {columnsection , nIp_stif} }
331     barsliptop = { string.format('columnbst_%d_%d',colbay , colfloor), 'InterfaceElement2D', node6, node7,
332         { {barslipspring , {1,0,0} } }, { {barslipsection} }, {0,1,0},{-1,0,0} }
333     rigidcolumnstop = { string.format('columnrdt_%d_%d',colbay , colfloor), 'StiffnessBased2DBeamColumn',
334         node7, node8, {columnrigidsection , nIp_elas} }
335     table.insert(elements , rigidcolumnbottom)
336     table.insert(elements , barslipbottom)
337     table.insert(elements , plasticcolumn1)
338     table.insert(elements , flexcolumn)
339     table.insert(elements , plasticcolumn2)
340     table.insert(elements , barsliptop)
341     table.insert(elements , rigidcolumnstop)
342 end
343 end
344 end
345
346 -- create beam node coordinates
347 for beambay = 1, nbays do
348     for beamfloor = 2, nflrs+1 do
349         y=cflen+colen*(beambay-1)
350         -- create nodes
351         for k=2, nbmnodes-1 do
352             x = (beambay-1)*bmlen + bmnod[k]
353             massbm= bmnodmass[k]
354             curnode = { beamnodename(beambay, beamfloor, k), x , y, 'mass', massbm, massbm, 0.1*massbm }
355             table.insert(nodes, curnode)
356         end
357         -- figure out section names
358         beamsection = 'BeamSection'
359         barslipsection = 'BSBeamSection'
360         beamrigidsection = 'BeamRigidSection'
361         barslipspring = 'BSBeamSS'
362         -- figure out node tag for node connectivity
363         node1 = beamnodename(beambay, beamfloor, 1)
364         node2 = beamnodename(beambay, beamfloor, 2)
365         node3 = beamnodename(beambay, beamfloor, 3)
366         node4 = beamnodename(beambay, beamfloor, 4)
367         node5 = beamnodename(beambay, beamfloor, 5)
368         node6 = beamnodename(beambay, beamfloor, 6)
369         node7 = beamnodename(beambay, beamfloor, 7)
370         node8 = beamnodename(beambay+1, beamfloor, 1)
371         -- Define elements
372         rigidbeamleft = { string.format('beamrdl_%d_%d', beambay, beamfloor),
373             'StiffnessBased2DBeamColumn', node1, node2, {beamrigidsection , nIp_elas} }
374         barslipleft = { string.format('beambsl_%d_%d', beambay, beamfloor),
375             'InterfaceElement2D', node2, node3,
376             { {barslipspring , {0,1,0} } }, { {barslipsection} }, {1,0,0},{0,1,0} }
377         plasticbeam1 = { string.format('beamp1l_%d_%d', beambay, beamfloor),
378             'StiffnessBased2DBeamColumn', node3, node4, {beamsection , nIp_stif} }
379         flexbeam = { string.format('beamflx_%d_%d', beambay, beamfloor),
380             'FlexibilityBased2DBeamColumn', node4, node5, {beamsection , nIp_flex}, flexparams }
381         plasticbeam2 = { string.format('beamp2l_%d_%d', beambay, beamfloor),
382             'StiffnessBased2DBeamColumn', node5, node6, {beamsection , nIp_stif} }
383         barslipright = { string.format('beambsr_%d_%d', beambay, beamfloor),
384             'InterfaceElement2D', node6, node7, { {barslipspring , {0,1,0} } },
385             { {barslipsection} }, {1,0,0},{0,1,0} }
386         rigidbeamright= { string.format('beamrdr_%d_%d', beambay, beamfloor), 'StiffnessBased2DBeamColumn',
387             node7, node8, {beamrigidsection , nIp_elas} }
388         table.insert(elements , rigidbeamleft)
389         table.insert(elements , barslipleft)
390         table.insert(elements , plasticbeam1)
391         table.insert(elements , flexbeam)
392         table.insert(elements , plasticbeam2)
393         table.insert(elements , barslipright)
394         table.insert(elements , rigidbeamright)
395     end
396 end
397 -- *****
398 -- Set section properties.
399 --
400 --
401 allsections = {
402 --
403 'ColDSection', 'Fiber',
404 -- Tag, Area, y-loc, z-loc
405 {
406     'ColDCover', 5.1, 2.575, 0,
407     'ColDCover', 5.1, -2.575, 0,
408     'ColDCover', 0.91375, 1.88125, 0,
409     'ColDCover', 0.91375, 1.34375, 0,
410     'ColDCover', 0.91375, 0.80625, 0,
411     'ColDCover', 0.91375, 0.26875, 0,
412     'ColDCover', 0.91375, -0.26875, 0,
413     'ColDCover', 0.91375, -0.80625, 0,
414     'ColDCover', 0.91375, -1.34375, 0,
415     'ColDCover', 0.91375, -1.88125, 0,
416     'ColDCore', 2.31125, 1.88125, 0,
417     'ColDCore', 2.31125, 1.34375, 0,
418     'ColDCore', 2.31125, 0.80625, 0,
419     'ColDCore', 2.31125, 0.26875, 0,
420     'ColDCore', 2.31125, -0.26875, 0,
421     'ColDCore', 2.31125, -0.80625, 0,
422     'ColDCore', 2.31125, -1.34375, 0,
423     'ColDCore', 2.31125, -1.88125, 0,
424     'ColDSteel', 0.147, 2.15, 0,
425     'ColDSteel', 0.098, 0, 0,
426     'ColDSteel', 0.147, -2.15, 0 };

```



```

426 -----
427 'BSColDSection' , 'Fiber' ,
428 { 'BSColDCover' , 5.1 , 2.575 , 0 ,
429   'BSColDCover' , 5.1 , -2.575 , 0 ,
430   'BSColDCover' , 0.91375 , 1.88125 , 0 ,
431   'BSColDCover' , 0.91375 , 1.34375 , 0 ,
432   'BSColDCover' , 0.91375 , 0.80625 , 0 ,
433   'BSColDCover' , 0.91375 , 0.26875 , 0 ,
434   'BSColDCover' , 0.91375 , -0.26875 , 0 ,
435   'BSColDCover' , 0.91375 , -0.80625 , 0 ,
436   'BSColDCover' , 0.91375 , -1.34375 , 0 ,
437   'BSColDCover' , 0.91375 , -1.88125 , 0 ,
438   'BSColDCore' , 2.31125 , 1.88125 , 0 ,
439   'BSColDCore' , 2.31125 , 1.34375 , 0 ,
440   'BSColDCore' , 2.31125 , 0.80625 , 0 ,
441   'BSColDCore' , 2.31125 , 0.26875 , 0 ,
442   'BSColDCore' , 2.31125 , -0.26875 , 0 ,
443   'BSColDCore' , 2.31125 , -0.80625 , 0 ,
444   'BSColDCore' , 2.31125 , -1.34375 , 0 ,
445   'BSColDCore' , 2.31125 , -1.88125 , 0 ,
446   'BSColDSteel' , 0.147 , 2.15 , 0 ,
447   'BSColDSteel' , 0.098 , 0 , 0 ,
448   'BSColDSteel' , 0.147 , -2.15 , 0 } ;
449 -----
450 'BSColDFSection' , 'Fiber' ,
451 { 'BSColDFCover' , 5.1 , 2.575 , 0 ,
452   'BSColDFCover' , 5.1 , -2.575 , 0 ,
453   'BSColDFCover' , 0.91375 , 1.88125 , 0 ,
454   'BSColDFCover' , 0.91375 , 1.34375 , 0 ,
455   'BSColDFCover' , 0.91375 , 0.80625 , 0 ,
456   'BSColDFCover' , 0.91375 , 0.26875 , 0 ,
457   'BSColDFCover' , 0.91375 , -0.26875 , 0 ,
458   'BSColDFCover' , 0.91375 , -0.80625 , 0 ,
459   'BSColDFCover' , 0.91375 , -1.34375 , 0 ,
460   'BSColDFCover' , 0.91375 , -1.88125 , 0 ,
461   'BSColDFCore' , 2.31125 , 1.88125 , 0 ,
462   'BSColDFCore' , 2.31125 , 1.34375 , 0 ,
463   'BSColDFCore' , 2.31125 , 0.80625 , 0 ,
464   'BSColDFCore' , 2.31125 , 0.26875 , 0 ,
465   'BSColDFCore' , 2.31125 , -0.26875 , 0 ,
466   'BSColDFCore' , 2.31125 , -0.80625 , 0 ,
467   'BSColDFCore' , 2.31125 , -1.34375 , 0 ,
468   'BSColDFCore' , 2.31125 , -1.88125 , 0 ,
469   'BSColDFSteel' , 0.147 , 2.15 , 0 ,
470   'BSColDFSteel' , 0.098 , 0 , 0 ,
471   'BSColDFSteel' , 0.147 , -2.15 , 0 } ;
472 -----
473 'ColNDSection' , 'Fiber' ,
474 { 'ColNDCover' , 5.1 , 2.575 , 0 ,
475   'ColNDCover' , 5.1 , -2.575 , 0 ,
476   'ColNDCover' , 0.91375 , 1.88125 , 0 ,
477   'ColNDCover' , 0.91375 , 1.34375 , 0 ,
478   'ColNDCover' , 0.91375 , 0.80625 , 0 ,
479   'ColNDCover' , 0.91375 , 0.26875 , 0 ,
480   'ColNDCover' , 0.91375 , -0.26875 , 0 ,
481   'ColNDCover' , 0.91375 , -0.80625 , 0 ,
482   'ColNDCover' , 0.91375 , -1.34375 , 0 ,
483   'ColNDCover' , 0.91375 , -1.88125 , 0 ,
484   'ColNDCore' , 2.31125 , 1.88125 , 0 ,
485   'ColNDCore' , 2.31125 , 1.34375 , 0 ,
486   'ColNDCore' , 2.31125 , 0.80625 , 0 ,
487   'ColNDCore' , 2.31125 , 0.26875 , 0 ,
488   'ColNDCore' , 2.31125 , -0.26875 , 0 ,
489   'ColNDCore' , 2.31125 , -0.80625 , 0 ,
490   'ColNDCore' , 2.31125 , -1.34375 , 0 ,
491   'ColNDCore' , 2.31125 , -1.88125 , 0 ,
492   'ColNDSteel' , 0.33 , 2.15 , 0 ,
493   'ColNDSteel' , 0.22 , 0 , 0 ,
494   'ColNDSteel' , 0.33 , -2.15 , 0 } ;
495 -----
496 'BSColNDSection' , 'Fiber' ,
497 { 'BSColNDCover' , 5.1 , 2.575 , 0 ,
498   'BSColNDCover' , 5.1 , -2.575 , 0 ,
499   'BSColNDCover' , 0.91375 , 1.88125 , 0 ,
500   'BSColNDCover' , 0.91375 , 1.34375 , 0 ,
501   'BSColNDCover' , 0.91375 , 0.80625 , 0 ,
502   'BSColNDCover' , 0.91375 , 0.26875 , 0 ,
503   'BSColNDCover' , 0.91375 , -0.26875 , 0 ,
504   'BSColNDCover' , 0.91375 , -0.80625 , 0 ,
505   'BSColNDCover' , 0.91375 , -1.34375 , 0 ,
506   'BSColNDCover' , 0.91375 , -1.88125 , 0 ,
507   'BSColNDCore' , 2.31125 , 1.88125 , 0 ,
508   'BSColNDCore' , 2.31125 , 1.34375 , 0 ,
509   'BSColNDCore' , 2.31125 , 0.80625 , 0 ,
510   'BSColNDCore' , 2.31125 , 0.26875 , 0 ,
511   'BSColNDCore' , 2.31125 , -0.26875 , 0 ,
512   'BSColNDCore' , 2.31125 , -0.80625 , 0 ,
513   'BSColNDCore' , 2.31125 , -1.34375 , 0 ,
514   'BSColNDCore' , 2.31125 , -1.88125 , 0 ,
515   'BSColNDSteel' , 0.33 , 2.15 , 0 ,
516   'BSColNDSteel' , 0.22 , 0 , 0 ,
517   'BSColNDSteel' , 0.33 , -2.15 , 0 } ;
518 -----
519 'BSColNDFSection' , 'Fiber' ,
520 { 'BSColNDFCover' , 5.1 , 2.575 , 0 ,
521   'BSColNDFCover' , 5.1 , -2.575 , 0 ,
522   'BSColNDFCover' , 0.91375 , 1.88125 , 0 ,
523   'BSColNDFCover' , 0.91375 , 1.34375 , 0 ,
524   'BSColNDFCover' , 0.91375 , 0.80625 , 0 ,
525   'BSColNDFCover' , 0.91375 , 0.26875 , 0 ,
526   'BSColNDFCover' , 0.91375 , -0.26875 , 0 ,
527   'BSColNDFCover' , 0.91375 , -0.80625 , 0 ,
528   'BSColNDFCover' , 0.91375 , -1.34375 , 0 ,
529   'BSColNDFCover' , 0.91375 , -1.88125 , 0 ,
530   'BSColNDFCore' , 2.31125 , 1.88125 , 0 ,
531   'BSColNDFCore' , 2.31125 , 1.34375 , 0 ,
532   'BSColNDFCore' , 2.31125 , 0.80625 , 0 ,

```

```

533 'BSCoINDFCore' , 2.31125 , 0.26875 , 0 ,
534 'BSCoINDFCore' , 2.31125 , -0.26875 , 0 ,
535 'BSCoINDFCore' , 2.31125 , -0.80625 , 0 ,
536 'BSCoINDFCore' , 2.31125 , -1.34375 , 0 ,
537 'BSCoINDFCore' , 2.31125 , -1.88125 , 0 ,
538 'BSCoINDFSteel' , 0.33 , 2.15 , 0 ,
539 'BSCoINDFSteel' , 0.22 , 0 , 0 ,
540 'BSCoINDFSteel' , 0.33 , -2.15 , 0 ;};
541
542 'BeamSection' , 'Fiber' ,
543 { 'BeamCover' , 5.1 , 4.075 , 0 ,
544 'BeamCover' , 5.1 , -4.075 , 0 ,
545 'BeamCover' , 1.55125 , 3.19375 , 0 ,
546 'BeamCover' , 1.55125 , 2.28125 , 0 ,
547 'BeamCover' , 1.55125 , 1.36875 , 0 ,
548 'BeamCover' , 1.55125 , 0.45625 , 0 ,
549 'BeamCover' , 1.55125 , -0.45625 , 0 ,
550 'BeamCover' , 1.55125 , -1.36875 , 0 ,
551 'BeamCover' , 1.55125 , -2.28125 , 0 ,
552 'BeamCover' , 1.55125 , -3.19375 , 0 ,
553 'BeamCore' , 3.92375 , 3.19375 , 0 ,
554 'BeamCore' , 3.92375 , 2.28125 , 0 ,
555 'BeamCore' , 3.92375 , 1.36875 , 0 ,
556 'BeamCore' , 3.92375 , 0.45625 , 0 ,
557 'BeamCore' , 3.92375 , -0.45625 , 0 ,
558 'BeamCore' , 3.92375 , -1.36875 , 0 ,
559 'BeamCore' , 3.92375 , -2.28125 , 0 ,
560 'BeamCore' , 3.92375 , -3.19375 , 0 ,
561 'BeamSteel' , 0.44 , 3.65 , 0 ,
562 'BeamSteel' , 0.44 , -3.65 , 0 ;};
563
564 'BSBeamSection' , 'Fiber' ,
565 { 'BSBeamCover' , 5.1 , 4.075 , 0 ,
566 'BSBeamCover' , 5.1 , -4.075 , 0 ,
567 'BSBeamCover' , 1.55125 , 3.19375 , 0 ,
568 'BSBeamCover' , 1.55125 , 2.28125 , 0 ,
569 'BSBeamCover' , 1.55125 , 1.36875 , 0 ,
570 'BSBeamCover' , 1.55125 , 0.45625 , 0 ,
571 'BSBeamCover' , 1.55125 , -0.45625 , 0 ,
572 'BSBeamCover' , 1.55125 , -1.36875 , 0 ,
573 'BSBeamCover' , 1.55125 , -2.28125 , 0 ,
574 'BSBeamCover' , 1.55125 , -3.19375 , 0 ,
575 'BSBeamCore' , 3.92375 , 3.19375 , 0 ,
576 'BSBeamCore' , 3.92375 , 2.28125 , 0 ,
577 'BSBeamCore' , 3.92375 , 1.36875 , 0 ,
578 'BSBeamCore' , 3.92375 , 0.45625 , 0 ,
579 'BSBeamCore' , 3.92375 , -0.45625 , 0 ,
580 'BSBeamCore' , 3.92375 , -1.36875 , 0 ,
581 'BSBeamCore' , 3.92375 , -2.28125 , 0 ,
582 'BSBeamCore' , 3.92375 , -3.19375 , 0 ,
583 'BSBeamSteel' , 0.44 , 3.65 , 0 ,
584 'BSBeamSteel' , 0.44 , -3.65 , 0 ;};
585
586 'ColRigidSection' , 'general' ,
587 { 'ColRigid' , 36 , 108 ;};
588
589 'BeamRigidSection' , 'general' ,
590 { 'BeamRigid' , 54 , 364.5 ;};
591
592 --- *****
593 --- Set material properties .
594 ---
595 ---
596 concretemat = 'ConcreteLinearTensionSoftening'
597 steelmat = 'GiuffreMenegottoPinto'
598 sspringmat = 'Bilinear'
599 materials = {
600 --- Concrete (ConcreteLinearTensionSoftening)
601 Tag, mat_type, E, density, sc, ec, scu, ecu, lam, st
602 {'ColDCover' , concretemat , 549.2307692 , 0 , -3.57 , -0.0026 , -1.19 , -0.0078 , 0.3 , 0.448121077 } ,
603 {'ColDCore' , concretemat , 549.4505495 , 0 , -7.5 , -0.00546 , -7.35 , -0.01638 , 0.3 , 0.649519053 } ,
604 {'BSCoIDCover' , concretemat , 105 , 0 , -3.57 , -0.0136 , -1.19 , -0.0408 , 0.3 , 0.448121077 } ,
605 {'BSCoIDCore' , concretemat , 104.1666667 , 0 , -7.5 , -0.0288 , -7.35 , -0.0864 , 0.3 , 0.649519053 } ,
606 {'BSCoIDFCover' , concretemat , 105 , 0 , -3.57 , -0.0136 , -1.19 , -0.0408 , 0.3 , 0.448121077 } ,
607 {'BSCoIDFCore' , concretemat , 104.1666667 , 0 , -7.5 , -0.0288 , -6.75 , -0.0864 , 0.3 , 0.649519053 } ,
608 {'ColNDCover' , concretemat , 549.2307692 , 0 , -3.57 , -0.0026 , -1.19 , -0.0078 , 0.3 , 0.448121077 } ,
609 {'ColNDCore' , concretemat , 549.2957746 , 0 , -3.9 , -0.00284 , -3.51 , -0.00852 , 0.3 , 0.46837485 } ,
610 {'BSCoINDCover' , concretemat , 105 , 0 , -3.57 , -0.0136 , -1.19 , -0.0408 , 0.3 , 0.448121077 } ,
611 {'BSCoINDCore' , concretemat , 106.8493151 , 0 , -3.9 , -0.0146 , -3.51 , -0.0438 , 0.3 , 0.46837485 } ,
612 {'BSCoINDFCover' , concretemat , 144.2424242 , 0 , -3.57 , -0.0099 , -1.19 , -0.0297 , 0.3 , 0.448121077 } ,
613 {'BSCoINDFCore' , concretemat , 106.8493151 , 0 , -3.9 , -0.0146 , -3.51 , -0.0438 , 0.3 , 0.46837485 } ,
614 {'BeamCover' , concretemat , 549.2307692 , 0 , -3.57 , -0.0026 , -1.19 , -0.0078 , 0.3 , 0.448121077 } ,
615 {'BeamCore' , concretemat , 549.2957746 , 0 , -3.9 , -0.00284 , -3.51 , -0.00852 , 0.3 , 0.46837485 } ,
616 {'BSBeamCover' , concretemat , 105 , 0 , -3.57 , -0.0136 , -1.19 , -0.0408 , 0.3 , 0.448121077 } ,
617 {'BSBeamCore' , concretemat , 106.8493151 , 0 , -3.9 , -0.0146 , -3.51 , -0.0438 , 0.3 , 0.46837485 } ,
618 --- Steel (GiuffreMenegottoPinto)
619 Tag, mat_type, E, density, sy, b, R0, cR1, cR2,a1, a2,a3, a4, s-init
620 {'ColDSteel' , steelmat , 26500 , 0 , 87.5 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
621 {'BSCoIDSteel' , steelmat , 5067 , 0 , 87.5 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
622 {'BSCoIDFSteel' , steelmat , 6949 , 0 , 87.5 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
623 {'ColNDSteel' , steelmat , 27300 , 0 , 80 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
624 {'BSCoINDSteel' , steelmat , 5220 , 0 , 80 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
625 {'BSCoINDFSteel' , steelmat , 5220 , 0 , 80 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
626 {'BeamSteel' , steelmat , 27300 , 0 , 80 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
627 {'BSBeamSteel' , steelmat , 5220 , 0 , 80 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
628 --- Shear spring (Bilinear)
629 Tag, mat_type, E, density, sy, b, a1, a2, a3, a4
630 {'BSCoIDSS' , sspringmat , 1690 , 0 , 78.2 , 0.173 , 0 , 55 , 0 , 55 } ,
631 {'BSCoIDFSS' , sspringmat , 1690 , 0 , 78.2 , 0.173 , 0 , 55 , 0 , 55 } ,
632 {'BSCoINDSS' , sspringmat , 1690 , 0 , 78.2 , 0.173 , 0 , 55 , 0 , 55 } ,
633 {'BSCoINDFSS' , sspringmat , 1690 , 0 , 78.2 , 0.173 , 0 , 55 , 0 , 55 } ,
634 {'BSBeamSS' , sspringmat , 1545 , 0 , 75.8 , 0.396 , 0 , 55 , 0 , 55 } ,
635 --- Rigid element (Elastic)
636 Tag, mat_type, E, density
637 {'ColRigid' , 'Elastic' , 10000000000 , 0 } ,
638 {'BeamRigid' , 'Elastic' , 10000000000 , 0 }
639 }

```

```

640 --- *****
641 function dumptable(x)
642     local result = '{'
643     for i,v in ipairs(x) do
644         if (type(v) == 'table') then
645             result = result .. dumptable(v)
646         else
647             result = result .. v .. ','
648         end
649     end
650     return result .. '}\n'
651 end
652 ---
653 ---print(dumptable(nodes))
654 ---print(dumptable(elements))
655 ---print(dumptable(materials))
656 ---print(dumptable(allsections))
657 --- *****
658 --- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
659 model = StructureModel(2,3)
660 --- Assign all input data to Mercury
661 model:addNodes(nodes)
662 model:addMaterials(materials)
663 model:addSections(allsections)
664 model:addElements(elements)
665 --- *****
666 --- constrain bottom nodes
667 for bay=1,nbays+1 do
668     model:constrainNode(columnnodename(bay,1,1),1,1,1)
669 end
670 --- *****
671 solver = NonlinearSolver(" initialstiffness", { displacementdeltatolerance=1e-5, iterations=100000})
672 --- *****
673 --- Monitoring node number
674 node06 = columnnodename(1,1,6);
675 node13 = columnnodename(1,2,7)
676 node20 = columnnodename(1,3,7)
677 node07 = columnnodename(1,1,7)
678 node14 = columnnodename(1,2,8)
679 node21 = columnnodename(1,3,8)
680 node48 = columnnodename(3,1,6)
681 node49 = columnnodename(3,1,7)
682 node01 = columnnodename(1,1,1)
683 node22 = columnnodename(2,1,1)
684 node43 = columnnodename(3,1,1)
685 node64 = columnnodename(4,1,1)
686 --- *****
687 disp_06_13_20 = {}
688 function disp1pertime(timestep)
689     dx06,dy06,dz06 = model:nodeDisplacements(node06)
690     dx13,dy13,dz13 = model:nodeDisplacements(node13)
691     dx20,dy20,dz20 = model:nodeDisplacements(node20)
692     table.insert(disp_06_13_20,dx06)
693     table.insert(disp_06_13_20,dy06)
694     table.insert(disp_06_13_20,dz06)
695     table.insert(disp_06_13_20,dx13)
696     table.insert(disp_06_13_20,dy13)
697     table.insert(disp_06_13_20,dz13)
698     table.insert(disp_06_13_20,dx20)
699     table.insert(disp_06_13_20,dy20)
700     table.insert(disp_06_13_20,dz20)
701 end
702 disp_07_14_21 = {}
703 function disp2pertime(timestep)
704     dx07,dy07,dz07 = model:nodeDisplacements(node07)
705     dx14,dy14,dz14 = model:nodeDisplacements(node14)
706     dx21,dy21,dz21 = model:nodeDisplacements(node21)
707     table.insert(disp_07_14_21,dx07)
708     table.insert(disp_07_14_21,dy07)
709     table.insert(disp_07_14_21,dz07)
710     table.insert(disp_07_14_21,dx14)
711     table.insert(disp_07_14_21,dy14)
712     table.insert(disp_07_14_21,dz14)
713     table.insert(disp_07_14_21,dx21)
714     table.insert(disp_07_14_21,dy21)
715     table.insert(disp_07_14_21,dz21)
716 end
717 disp_48_49 = {}
718 function disp3pertime(timestep)
719     dx48,dy48,dz48 = model:nodeDisplacements(node48)
720     dx49,dy49,dz49 = model:nodeDisplacements(node49)
721     table.insert(disp_48_49,dx48)
722     table.insert(disp_48_49,dy48)
723     table.insert(disp_48_49,dz48)
724     table.insert(disp_48_49,dx49)
725     table.insert(disp_48_49,dy49)
726     table.insert(disp_48_49,dz49)
727 end
728 ---
729 react = {}
730 function reactpertime(timestep)
731     fx01,fy01,fz01 = model:nodeRestoringForces(node01)
732     fx22,fy22,fz22 = model:nodeRestoringForces(node22)
733     fx43,fy43,fz43 = model:nodeRestoringForces(node43)
734     fx64,fy64,fz64 = model:nodeRestoringForces(node64)
735     table.insert(react,fx01)
736     table.insert(react,fy01)
737     table.insert(react,fz01)
738     table.insert(react,fx22)
739     table.insert(react,fy22)
740     table.insert(react,fz22)
741     table.insert(react,fx43)
742     table.insert(react,fy43)
743     table.insert(react,fz43)
744     table.insert(react,fx64)
745     table.insert(react,fy64)
746     table.insert(react,fz64)

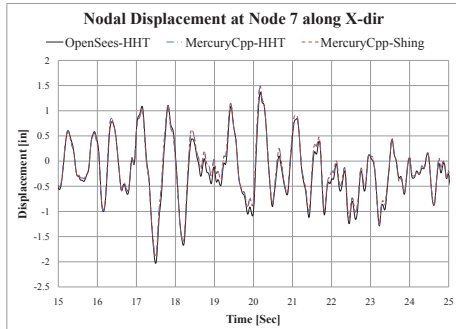
```

```

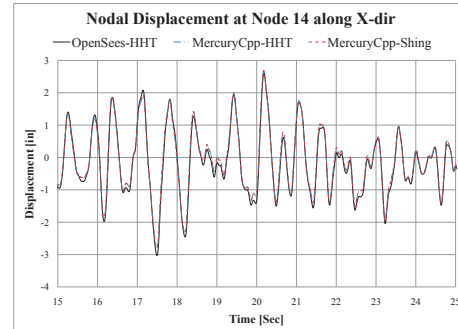
747     print(timestep);
748 end
749 ---
750 --- *****
751 earthquakeloading = LoadDescription ()
752 accelamp = 1.0*g
753 earthquakeloading.addLoad({'groundmotion', earthquakefile .. ',dt=0.005', 1, accelamp})
754 --- *****
755 ---transientanalysis = DynamicAnalysis("hht", model, solver, earthquakeloading,
756 ---[[ dt, alpha, beta, gamma]] 0.005, alpha, beta, gamma)
757 transientanalysis = DynamicAnalysis("hybridhht", model, solver, earthquakeloading,
758 ---[[ dt, alpha, beta, gamma, iteration]] 0.005, alpha, beta, gamma, 10)
759 model.setRayleighCoefficients(1.1770500887303603, 0.0015992014979696392)
760 --- *****
761 transientanalysis.addcallback(displpertime, "timestep")
762 transientanalysis.addcallback(disppertime, "timestep")
763 transientanalysis.addcallback(dispspertime, "timestep")
764 transientanalysis.addcallback(reactpertime, "timestep")
765 --- *****
766 ---transientanalysis.solve(15281)
767 transientanalysis.solve(15281)
768 --- *****
769 --- Set output file
770 function writedata6(x, fname)
771     local f = assert(io.open(fname, 'w'))
772     local writenl = 0
773     for i,v in ipairs(x) do
774         f:write(v, " ")
775         writenl = writenl + 1
776         --- length of row size: writenl
777         if (writenl > 5) then
778             writenl = 0
779             f:write("\n")
780         end
781     end
782     f:close()
783 end
784 ---
785 function writedata9(x, fname)
786     local f = assert(io.open(fname, 'w'))
787     local writenl = 0
788     for i,v in ipairs(x) do
789         f:write(v, " ")
790         writenl = writenl + 1
791         --- length of row size: writenl
792         if (writenl > 8) then
793             writenl = 0
794             f:write("\n")
795         end
796     end
797     f:close()
798 end
799 ---
800 function writedata12(x, fname)
801     local f = assert(io.open(fname, 'w'))
802     local writenl = 0
803     for i,v in ipairs(x) do
804         f:write(v, " ")
805         writenl = writenl + 1
806         --- length of row size: writenl
807         if (writenl > 11) then
808             writenl = 0
809             f:write("\n")
810         end
811     end
812     f:close()
813 end
814 --- *****
815 ---writedata9(disp_06_13_20, 'Ex29HHT_06_13_20_displ.dat')
816 ---writedata9(disp_07_14_21, 'Ex29HHT_07_14_21_displ.dat')
817 ---writedata6(disp_48_49, 'Ex29HHT_48_49_displ.dat')
818 ---writedata12(react, 'Ex29HHT_01_22_43_64_react.dat')
819
820 writedata9(disp_06_13_20, 'Ex29Shing_06_13_20_displ.dat')
821 writedata9(disp_07_14_21, 'Ex29Shing_07_14_21_displ.dat')
822 writedata6(disp_48_49, 'Ex29Shing_48_49_displ.dat')
823 writedata12(react, 'Ex29Shing_01_22_43_64_react.dat')
824 --- *****

```

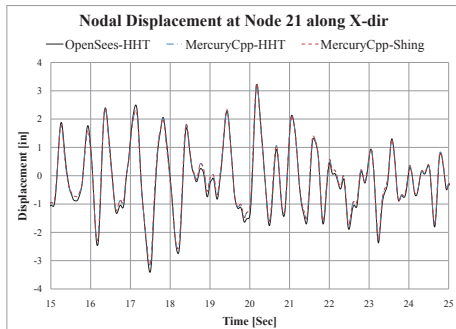
Fig. 17 shows the comparison for Mercury C++ and OpenSees.



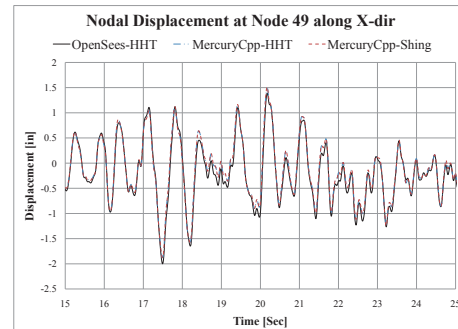
(a) Node 7



(b) Node 14



(c) Node 21



(d) Node 49

Figure 17: The comparison of displacement along X-dir at each node for Mercury C++ and OpenSees