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

List of Figures

List of Tables

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(disp):	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

```
%==========================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written    : September 2009.
% File name: Ex10.m
%==========================================================
%----------------------------------------------------------
% Description
% 1. Static and transient analysis
% 2. Load control
% 3. Iterative method
% 4. Simple 2D truss element
% 5. General section
% 6. Elastic and modified Giuffre-Monegotto-Pinto material
%----------------------------------------------------------
% Select type of analysis.
% AnalysisTyep = 1: Static analysis
% AnalysisType = 2: Transient analysis
AnalysisType = 2;
%----------------------------------------------------------
% Preface
Unit     = {'kN', 'mm'};
%          ndim, ndofpn
StrMode = {2, 2};
%----------------------------------------------------------
% Control block
Iteration = {'static',{ {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
                      }
             'transient',{ {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
                           {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
                           {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
                      }
            };
if (AnalysisType == 2)
    Integration = {'HHT', 0, -0.1, 0.3025, 0.6, 0, 0};
    eigens      = {0.02, 0.02};
end
%----------------------------------------------------------
% Geometry block
%           nodtag, x, y
nodcoord   = {1,     0, 0;
              2, 1500, 0;
              3, 3000, 0;
              4, 1500, 2000;
              5, 3000, 2000};
%           nodtag, x, y
```

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

Document Section	ID	Constitutive Model	Hardening	Static	Transient
2	Ex10	Elastic and ModGMP	Kinematic	Load control	HHT
3	Ex11	Damage		Cyclic disp control	
	Ex12	ModKP			
4	Ex16	Simplified Bilinear	No		
	Ex17	Simplified Bilinear	Isotropic		
	Ex18	ModGMP	Kinematic		
	Ex19	ModGMP	Kinematic & Isotropic		
5	Ex20	Classical Hardening	Isotropic		HHT
	Ex21		Kinematic		Shing
	Ex22		Kinematic & Isotropic		
6	Ex23	Elastic and Simplified Bilinear	No	Load control	
7	Ex24	Elastic and Bilinear	Isotropic		
8	Ex25	Classical Hardening		Pushover	
	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 wihtout 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(disp): Displacement.

Table 1: Features of Example Problems

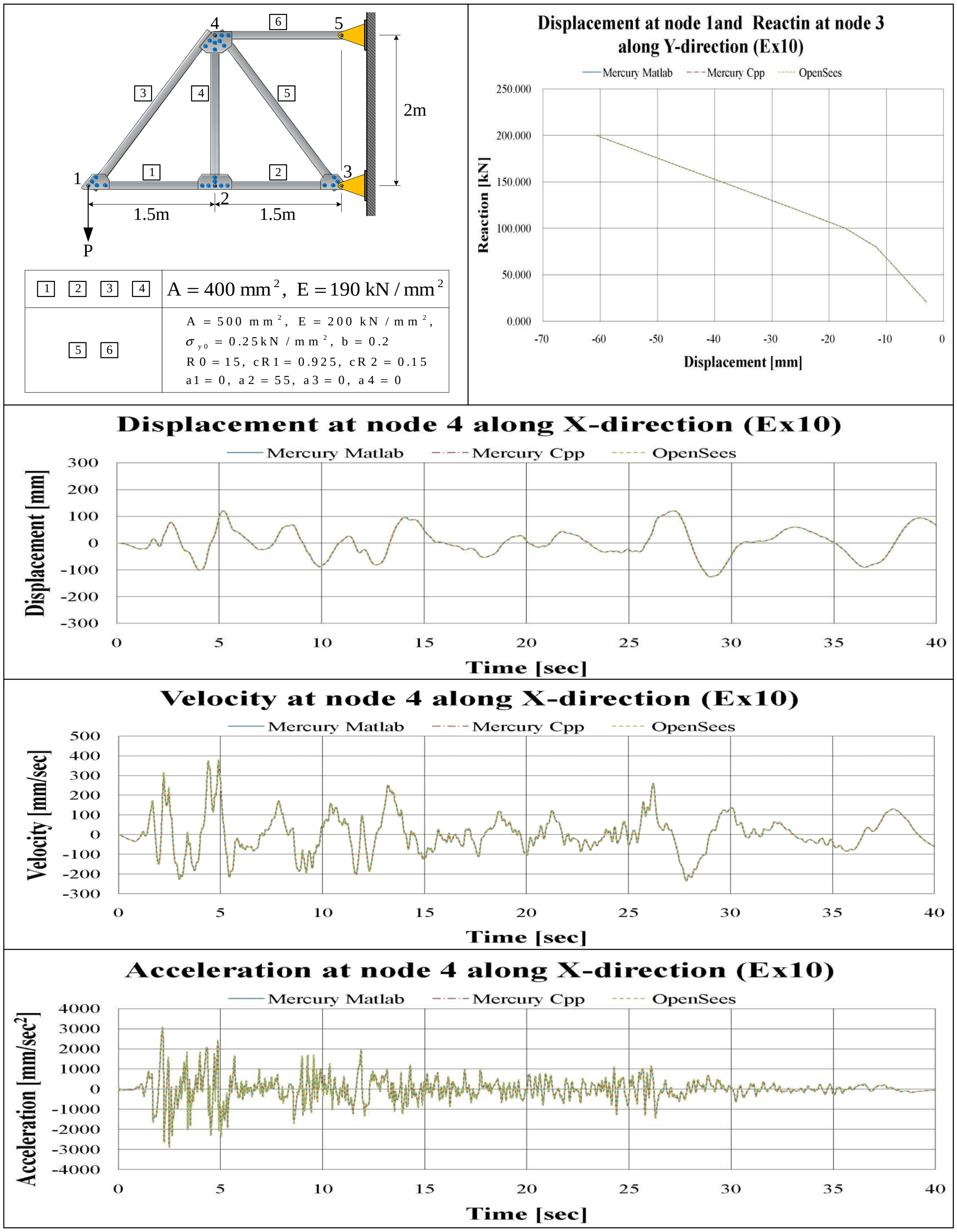


Figure 1: Ex10: Truss, ModGMP material, load control, HHT

```
constraint = {3, 1, 1;
              5, 1, 1};
%---------------------------------------------------------------------
% Element block
%           {eletag, 'Simple2DTruss', in, jn, sectag}
elements = { {1, 'Simple2DTruss', 1, 2, 1};
             {2, 'Simple2DTruss', 2, 3, 1};
             {3, 'Simple2DTruss', 1, 4, 1};
             {4, 'Simple2DTruss', 2, 4, 1};
             {5, 'Simple2DTruss', 3, 4, 2};
             {6, 'Simple2DTruss', 4, 5, 2} };
%---------------------------------------------------------------------
% Section block
%            sectag, 'General', {mattag, A, Ix, Iy, Iz}
sections = { 1, 'General', {1, 400, 0, 0, 0};
             2, 'General', {2, 500, 0, 0, 0}};
%---------------------------------------------------------------------
% Material block
%             mattag, 'Elastic', E, G, density
materials = { {1, 'Elastic', 190, 0, 7850*10^-9}
%             mattag, 'ModGMP', E, sy, b, R0, cR1, cR2, density, a1, a2, a3, a4
              {2, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 7850*10^-9, 0, 55, 0, 55} };
%---------------------------------------------------------------------
% Force block
if (AnalysisType == 1)
    forces = { 1, 'Static', {'NodalForces', {1, 2, -20}};
               2, 'LoadCtrl', {1, 2, {-40, -60, -80, -100, -120, -140, -160, -180, -200}} };
elseif (AnalysisType == 2)
    ga = load('ElCentro_g_0_01_Matlab.txt');
    nga = size(ga, 1);
    for i = 1:nga
        groundacceleration{i,1} = ga(i,1);
        groundacceleration{i,2} = ga(i,2);
        groundacceleration{i,3} = ga(i,3);
    end
    forces = { 1, 'Static', {'NodalForces', {1, 2, 0} };
               2, 'Acceleration', {9810, groundacceleration} };
end
%------------------------------------------------------------
```

2.2 C++

```
-- *******************************************************
-- AnalysisType = 1: Static analysis
-- AnalysisType = 2: Transient analysis
AnalysisType = 2
-- *******************************************************
%nodes = { {1, 0, 0, 'mass', 6.28, 6.28},
          {2, 1500, 0, 'mass', 7.85, 7.85},
          {3, 3000, 0},
          {4, 1500, 2000, 'mass', 14.915, 14.915},
          {5, 3000, 2000} }
--
Simple2DTruss = 'truss2d'
elements = { {1, Simple2DTruss, 1, 2, 400, 1},
             {2, Simple2DTruss, 2, 3, 400, 1},
             {3, Simple2DTruss, 1, 4, 400, 1},
             {4, Simple2DTruss, 2, 4, 400, 1},
             {5, Simple2DTruss, 3, 4, 500, 2},
             {6, Simple2DTruss, 4, 5, 500, 2} }
--{tag, 'modifiedGMPsteel', E, rho, fy, b_ratio, R0, cR1, cR2, a1, a2, a3, a4, sigma_init}
materials = { {1,'elastic',190,0.0};
              {2,'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 0, 55, 0, 55, 0} }
-- *******************************************************
model = StructureModel(2,2)
model:addNodes(nodes)
model:addMaterials(materials)
model:addElements(elements)
model:constrainNode(3,1,1)
model:constrainNode(5,1,1)
-- *******************************************************
-- Static analysis
if (AnalysisType == 1) then
    print("Static analysis started\n")
    staticloading = LoadDescription()
    staticloading:addLoad({'incrementalnodalload', 1, 2, -20,-40,-60,-80,-100,-120,-140,-160,-180,-200})
    -- *******************************************************
    displ = {}
    function displperstep(increment)
      dx1,dy1 = model:nodeDisplacements(1)
        dx2,dy2 = model:nodeDisplacements(2)
        dx4,dy4 = model:nodeDisplacements(4)
        table.insert(displ, dx1)
        table.insert(displ, dy1)
        table.insert(displ, dx2)
        table.insert(displ, dy2)
        table.insert(displ, dx4)
        table.insert(displ, dy4)
    end
    --
    react = {}
    function reactperstep(increment)
        fx3,fy3 = model:nodeRestoringForces(3)
        fx5,fy5 = model:nodeRestoringForces(5)
        table.insert(react, fx3)
        table.insert(react, fy3)
        table.insert(react, fx5)
        table.insert(react, fy5)
    end
    --
    solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
    analysis = StaticAnalysis(solver)
    analysis:setStructureModel(model)
```

```
    analysis:addcallback(displperstep,"increment")
    analysis:addcallback(reactperstep,"increment")
    analysis:solve(staticloading)
    -- *******************************************************
    -- Set output file
    function writedata6(x, fname)
        local f = assert(io.open(fname,'w'))
        local writenl = 0
        for i,v in ipairs(x) do
            f:write(v, " ")
            writenl = writenl + 1
            -- length of row size: writenl
            if (writenl > 5) then
                writenl = 0
                f:write("\n")
            end
        end
        f:close()
    end
    --
    function writedata4(x, fname)
        local f = assert(io.open(fname,'w'))
        local writenl = 0
        for i,v in ipairs(x) do
            f:write(v, " ")
            writenl = writenl + 1
            -- length of row size: writenl
            if (writenl > 3) then
                writenl = 0
                f:write("\n")
            end
        end
        f:close()
    end
    --
    writedata6(displ,'Ex10StaticNodalDisp_1_2_4.dat')
    writedata4(react,'Ex10StaticReact_3_5.dat')
    print("Static analysis ended\n")
end
-- *******************************************************
if (AnalysisType == 2) then
    print("Transient analysis started\n")
    earthquakeloading = LoadDescription()
    accelamp = 9810
    earthquakeloading:addLoad({'groundmotion','ElCentro_g_0_01_OpenSees.txt,dt=0.01', 1,accelamp})
    -- *******************************************************
    displ = {}
    function displpertime(time)
        dx1,dy1 = model:nodeDisplacements(1)
        dx2,dy2 = model:nodeDisplacements(2)
        dx4,dy4 = model:nodeDisplacements(4)
        table.insert(displ, dx1)
        table.insert(displ, dy1)
        table.insert(displ, dx2)
        table.insert(displ, dy2)
        table.insert(displ, dx4)
        table.insert(displ, dy4)
    end
  --
    react = {}
    function reactpertime(time)
        fx3,fy3 = model:nodeRestoringForces(3)
        fx5,fy5 = model:nodeRestoringForces(5)
        table.insert(react, fx3)
        table.insert(react, fy3)
        table.insert(react, fx5)
        table.insert(react, fy5)
    end
  --
    veloc = {}
    function velocpertime(time)
        vx1,vy1 = model:nodeVelocities(1)
        vx2,vy2 = model:nodeVelocities(2)
        vx4,vy4 = model:nodeVelocities(4)
        table.insert(veloc, vx1)
        table.insert(veloc, vy1)
        table.insert(veloc, vx2)
        table.insert(veloc, vy2)
        table.insert(veloc, vx4)
        table.insert(veloc, vy4)
    end
  --
    accel = {}
    function accelpertime(time)
        ax1,ay1 = model:nodeAccelerations(1)
        ax2,ay2 = model:nodeAccelerations(2)
        ax4,ay4 = model:nodeAccelerations(4)
        table.insert(accel, ax1)
        table.insert(accel, ay1)
        table.insert(accel, ax2)
        table.insert(accel, ay2)
        table.insert(accel, ax4)
        table.insert(accel, ay4)
    end
    -- *******************************************************
    solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-6, iterations=10})
  transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.1, 0.3025, 0.6)
    transientanalysis:addcallback(displpertime, "timestep")
    transientanalysis:addcallback(reactpertime, "timestep")
    transientanalysis:addcallback(velocpertime, "timestep")
    transientanalysis:addcallback(accelpertime, "timestep")
    model:setRayleighCoefficients(0.025533,0.007826)
    transientanalysis:solve(4000)
    -- *******************************************************
    function writedata6(x, fname)
        local f = assert(io.open(fname,'w'))
        local writenl = 0
```

```
            for i,v in ipairs(x) do
                f:write(v, " ")
                writenl = writenl + 1
                -- length of row size: writenl
                if (writenl > 5) then
                    writenl = 0
                    f:write("\n")
                end
            end
            f:close()
        end
        --
        function writedata4(x, fname)
            local f = assert(io.open(fname,'w'))
            local writenl = 0
            for i,v in ipairs(x) do
                f:write(v, " ")
                writenl = writenl + 1
                -- length of row size: writenl
                if (writenl > 3) then
                    writenl = 0
                    f:write("\n")
                end
            end
            f:close()
        end
        --
        writedata6(displ,'Ex10TransientNodalDisp_1_2_4.dat')
        writedata4(react,'Ex10TransientReact_3_5.dat')
        writedata6(veloc,'Ex10TransientNodalVelo_1_2_4.dat')
        writedata6(accel,'Ex10TransientNodalAcce_1_2_4.dat')
        print("Transient analysis ended\n")
    end
    -- ******************************************************
```

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.

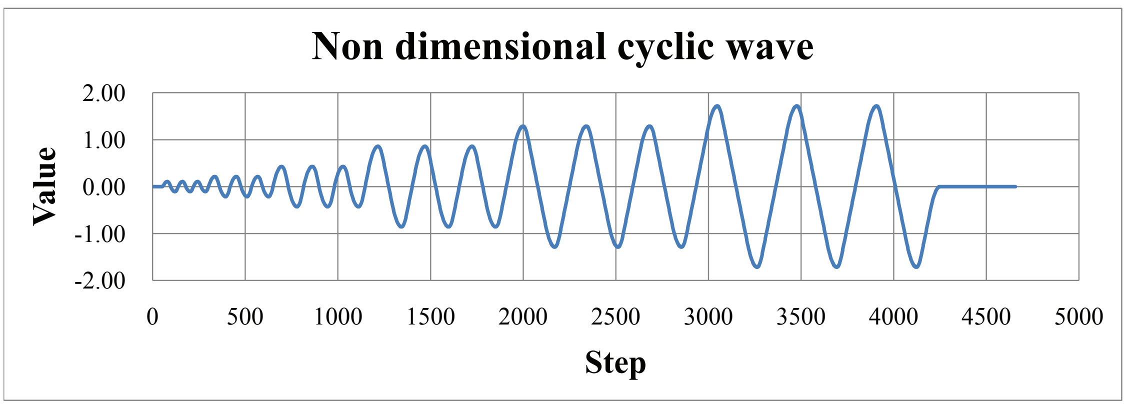


Figure 2: Cyclic wave without dimension

3.1 MATLAB

```
%==============================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written    : November 2009.
% File name: Ex11to12.m
%==============================================================
%--------------------------------------------------------------
% Description
% 1. Static analysis
% 2. Displacement control
% 3. Iterative method
% 4. Simple 2D truss element
% 5. General section
% 6. Anisotropic damage material(Ex11)
%    and modified Kent-Park material(Ex12)
%--------------------------------------------------------------
% Select material type
% MatType = 1: anisotropic damage material with permanant strain (Ex11)
% MatType = 2: modified Kent-Park material (Ex12)
MatType = 1;
%--------------------------------------------------------------
```

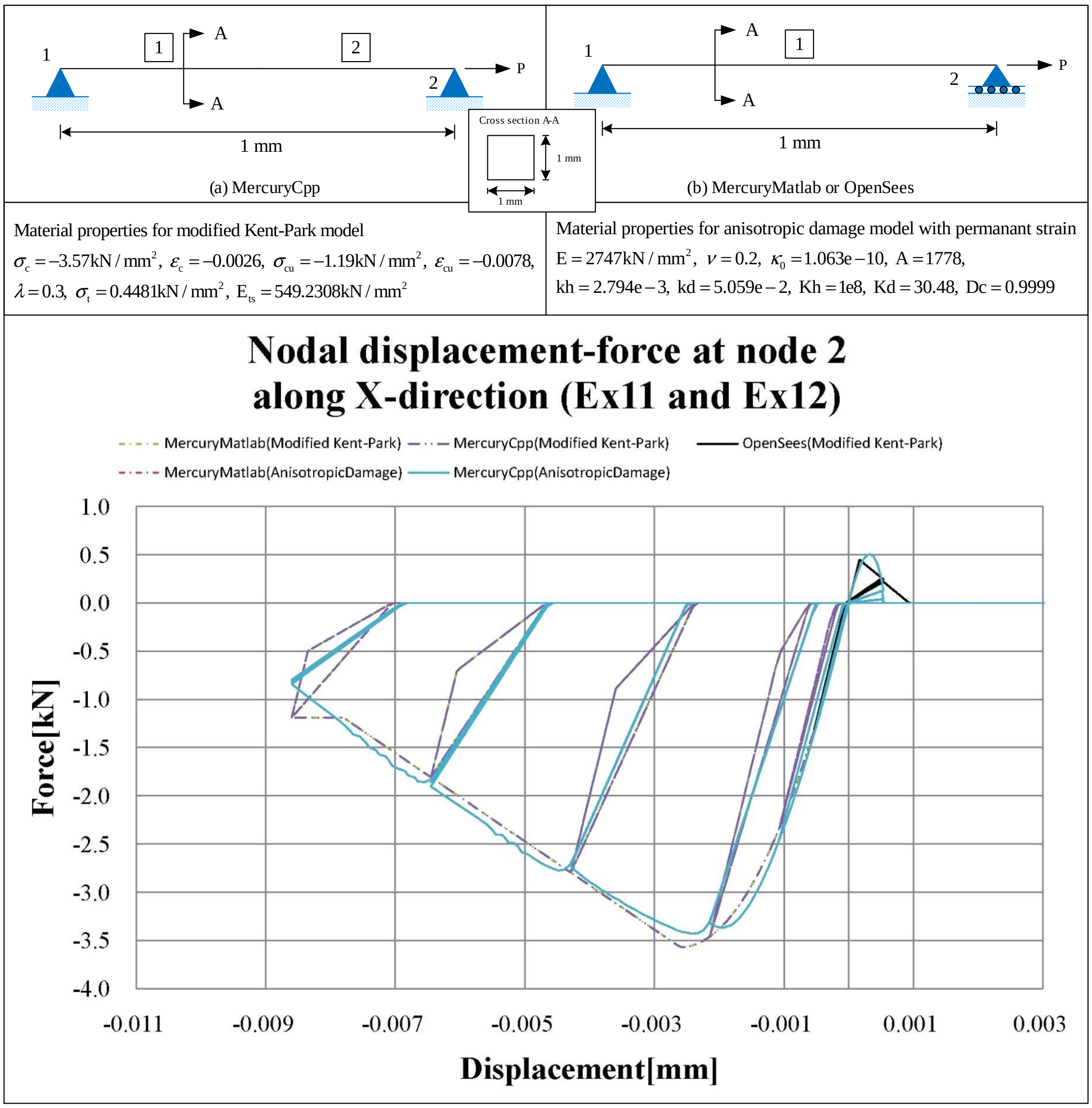


Figure 3: Examples 11- 12

```
% Preface
Unit    = {'kN', 'mm'};
StrMode = {2, 2};
%------------------------------------------------------------
% Control block
Iteration = {'static',{ {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
                      }
            };
%------------------------------------------------------------
% Geometry block
nodcoord   = {1,   0, 0;
              2,   1, 0;};
constraint = {1, 1, 1;
              2, 0, 1};
%------------------------------------------------------------
% Element block
elements = { {1, 'Simple2DTruss', 1, 2, 1} };
%------------------------------------------------------------
% Section block
sections = { 1, 'General', {1, 1, 0, 0, 0}};
%------------------------------------------------------------
% Material block
if (MatType == 1)
    % {tag, 'AnisotropicDamage', E, nu, kappa0, A, kh, kd, Kh, Kd, Dc, density};}
    materials = { {1, 'AnisotropicDamage', 2.747e+003, 0.2, 1.063e-010, 1.778e+003,
                   2.794e-003, 5.059e-002, 1.000e+008, 3.048e+001, 0.99999999, 0 } };
end
if (MatType == 2)
    % {tag, 'ModKP', sc, ec, scu, ecu, lambda, st, Ets, density};
    materials = {{1, 'ModKP', -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077, 549.2307692, 0}};
end
%------------------------------------------------------------
% Force block
DispInput = load('cyclicwave.txt');
row = size(DispInput,1);
for i = 1:row
    DispCell{i} = 0.005*DispInput(i);
end
%          forcetag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
forces = { 1, 'Static', {'NodalForces', {2, 1, 0}};
           2, 'DispCtrl', {2, 1, DispCell } };
%------------------------------------------------------------
```

3.2 C++

```
-- ***********************************************************
-- MatType = 1: Anisotropic damage model with permanant strain (Ex11)
-- MatType = 2: Modified Kent-Park model (Ex12)
MatType = 2
-- ***********************************************************
nodes = { {1,   0, 0},
          {2,   1, 0} }
-- ***********************************************************
Simple2DTruss = 'truss2d'
elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
-- ***********************************************************
if (MatType == 1) then
    materials = {{1, 'anisotropicdamage2', 2.747e+003, 0, 0.2, 1.063e-010, 1.778e+003,
                  2.794e-003, 5.059e-002, 1.000e+008, 3.048e+001, 0.99999999} };
end
if (MatType == 2) then
        concretemat = 'ConcreteLinearTensionSoftening'
    materials = {{1,concretemat, 549.2307692, 0, -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077}};
end
-- ***********************************************************
model = StructureModel(2,2)
model:addNodes(nodes)
model:addMaterials(materials)
model:addElements(elements)
model:constrainNode(1,1,1)
model:constrainNode(2,1,1)
-- ***********************************************************
function generateincrementalload()
 -- format:                    tag                         node dof
 local loadform = {'incrementalnodaldisplacement', 2  ,  1}
 local f = assert(io.open('cyclicwave.txt','r'))
 local n = f:read("*number")
 while (n ~= nil) do
  table.insert(loadform, 0.005*n)
  n = f:read("*number")
 end
 f:close()
 return loadform
end
 -- ***********************************************************
staticloading = LoadDescription()
-- format: 'staticnodalload' <node> <dof> <amplitude>
l = generateincrementalload()
staticloading:addLoad(l)
-- ***********************************************************
-- Static analysis
print("Static analysis started\n")
displ = {}
function displperstep(increment)
    dx2,dy2 = model:nodeDisplacements(2)
    table.insert(displ, dx2)
end
--
react = {}
function reactperstep(increment)
    fx1,fy1 = model:nodeRestoringForces(1)
```

```
    table.insert(react, fx1)
end
--
solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
analysis = StaticAnalysis(solver)
analysis:setStructureModel(model)
--analysis:addcallback(displperstep,"increment")
--analysis:addcallback(reactperstep,"increment")
analysis:solve(staticloading)
-- ***************************************************
-- Set output file
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
--
if (MatType == 1) then
    writedata1(displ,'Ex11StaticDamageNodalDisp_2.dat')
    writedata1(react,'Ex11StaticDamageReact_1.dat')
end
if (MatType == 2) then
    writedata1(displ,'Ex12StaticModKPNodalDisp_2.dat')
    writedata1(react,'Ex12StaticModKPReact_1.dat')
end
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

```
%==========================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written  : October 2009.
% File name: Ex16to17.m
%==========================================================
% Description
% 1. Static analysis
% 2. Displacement control
% 3. Iterative method
% 4. Simple 2D truss element
% 5. General sections
% 6. Bilinear material
%----------------------------------------------------------
% A36 Steel properties
% Density of 7.8 g/cm^3
% Minimum yield strength of 250 MPa(0.25 GPa)
% Ultimate tensile strength of 400-550 MPa(0.4-0.55 GPa)
%----------------------------------------------------------
% Select material type
% MatType = 1: Bilinear matierial without isotropic hardening
% MatType = 2: Bilinear matierial with isotropic hardening
MatType = 2;
%----------------------------------------------------------
% Preface
Unit    = {'kN', 'mm'};
StrMode = {2, 2};
%----------------------------------------------------------
% Control block
Iteration = {'static',{ {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
                        }};
%----------------------------------------------------------
% Geometry block
nodcoord   = {1, 0, 0;
              2, 1, 0};
constraint = {1, 1, 1;
              2, 0, 1};
%----------------------------------------------------------
% Element block
elements = { {1, 'Simple2DTruss', 1, 2, 1}; };
%----------------------------------------------------------
% Section block
sections = { 1, 'General', {1, 1, 0, 0, 0}};
%----------------------------------------------------------
% Material block
if MatType == 1
    materials = { {1, 'Bilinear', 200, 0.25, 0.2, 0, 0, 55, 0, 55} };
```

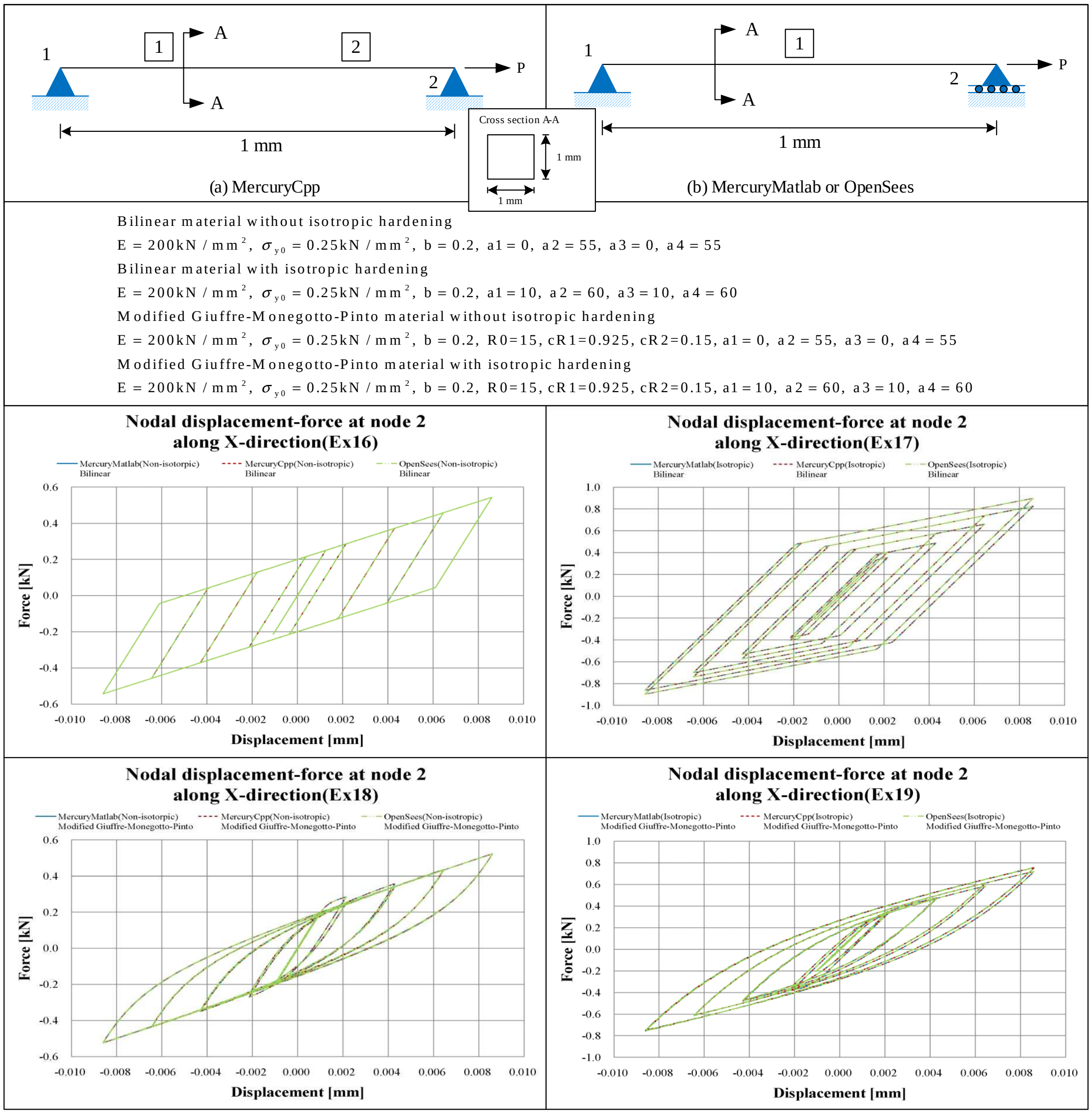


Figure 4: Examples 16- 19

```
elseif MatType == 2
    materials = { {1, 'Bilinear', 200, 0.25, 0.2, 0, 10, 60, 10, 60} };
end
%--------------------------------------------------------------------------
% Force block
DispInput = load('cyclicwave.txt');
row = size(DispInput,1);
for i = 1:row
    DispCell{i} = 0.005*DispInput(i);
end
%          forcetag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
forces = { 1, 'Static', {'NodalForces', {2, 1, 0}};
           2, 'DispCtrl', {2, 1, DispCell } };
%------------------------------------------------------------
```

4.2 C++

```
-- *******************************************************
-- MatType = 1: Bilinear material without isotropic hardening (Ex16)
-- MatType = 2: Bilinear material with isotropic hardening (Ex17)
MatType = 2
-- *******************************************************
nodes = { {1,    0, 0},
          {2,    1, 0} }
-- *******************************************************
Simple2DTruss = 'truss2d'
elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
-- *******************************************************
if (MatType == 1) then
-- materials = {{tag, 'bilinear', E, rho, fy, b_ratio, a1 ,a2, a3, a4}}
    materials = { {1,'bilinear', 200, 0, 0.25, 0.2, 0, 55, 0, 55} }
end
if (MatType == 2) then
    materials = { {1,'bilinear', 200, 0, 0.25, 0.2, 10, 60, 10, 60} }
end
-- *******************************************************
model = StructureModel(2,2)
model:addNodes(nodes)
model:addMaterials(materials)
model:addElements(elements)
model:constrainNode(1,1,1)
model:constrainNode(2,1,1)
-- *******************************************************
function generateincrementalload()
 -- format:              tag                          node dof
 local loadform = {'incrementalnodaldisplacement', 2 , 1}
 local f = assert(io.open('cyclicwave.txt','r'))
 local n = f:read("*number")
 while (n ~= nil) do
  table.insert(loadform, 0.005*n)
  n = f:read("*number")
 end
 f:close()
 return loadform
end
 -- *******************************************************
staticloading = LoadDescription()
-- format: 'staticnodalload' <node> <dof> <amplitude>
l = generateincrementalload()
staticloading:addLoad(l)
-- *******************************************************
-- Static analysis
print("Static analysis started\n")
displ = {}
function displperstep(increment)
    dx2,dy2 = model:nodeDisplacements(2)
    table.insert(displ, dx2)
end
--
react = {}
function reactperstep(increment)
    fx1,fy1 = model:nodeRestoringForces(1)
    table.insert(react, fx1)
end
--
solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
analysis = StaticAnalysis(solver)
analysis:setStructureModel(model)
analysis:addcallback(displperstep,"increment")
analysis:addcallback(reactperstep,"increment")
analysis:solve(staticloading)
-- *******************************************************
-- Set output file
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
--
if (MatType == 1) then
    writedata1(displ,'Ex16StaticNonIsotropicNodalDisp_2.dat')
    writedata1(react,'Ex16StaticNonIsotropicReact_1.dat')
end
if (MatType == 2) then
```

```
    writedata1(displ,'Ex17StaticIsotropicNodalDisp_2.dat')
    writedata1(react,'Ex17StaticIsotropicReact_1.dat')
end
print("Static analysis ended\n")
```

4.3 MATLAB

```
%================================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written  : October 2009.
% File name: Ex18to19.m
%================================================================
% Description
% 1. Static analysis
% 2. Displacement control
% 3. Iterative method
% 4. Simple 2D truss element
% 5. General sections
% 6. Modified Giuffre-Monegotto-Pinto material
%----------------------------------------------------------------
% A36 Steel properties
% Density of 7.8 g/cm^3
% Minimum yield strength of 250 MPa(0.25 GPa)
% Ultimate tensile strength of 400-550 MPa(0.4-0.55 GPa)
%----------------------------------------------------------------
% Select material type
% MatType = 1: ModGMP matierial without isotropic hardening (Ex18)
% MatType = 2: ModGMP matierial with isotropic hardening (Ex19)
MatType = 2;
%----------------------------------------------------------------
% Preface
Unit    = {'kN', 'mm'};
StrMode = {2, 2};
%----------------------------------------------------------------
% Control block
Iteration = {'static',{ {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
                        }};
%----------------------------------------------------------------
% Geometry block
nodcoord   = {1, 0, 0;
              2, 1, 0};
constraint = {1, 1, 1;
              2, 0, 1};
%----------------------------------------------------------------
% Element block
elements = { {1, 'Simple2DTruss', 1, 2, 1}; };
%----------------------------------------------------------------
% Section block
sections = { 1, 'General', {1, 1, 0, 0, 0}};
%----------------------------------------------------------------
% Material block
if MatType == 1
%         mattag, 'ModGMP', E, sy, b, R0, cR1, cR2, density, a1, a2, a3, a4
    materials = { {1, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 0, 0, 55, 0, 55} };
elseif MatType == 2
    materials = { {1, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 0, 10, 60, 10, 60} };
end
%----------------------------------------------------------------
% Force block
DispInput = load('cyclicwave.txt');
row = size(DispInput,1);
for i = 1:row
    DispCell{i} = 0.005*DispInput(i);
end
%          forcetag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
forces = { 1, 'Static', {'NodalForces', {2, 1, 0}};
           2, 'DispCtrl', {2, 1, DispCell } };
%--------------------------------------------------
```

4.4 C++

```
-- *******************************************************
-- MatType = 1: ModGMP material without isotropic hardening (Ex18)
-- MatType = 2: ModGMP material with isotropic hardening (Ex19)
MatType = 2
-- *******************************************************
nodes = { {1,   0, 0},
          {2,   1, 0} }
-- *******************************************************
Simple2DTruss = 'truss2d'
elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
-- *******************************************************
if (MatType == 1) then
-- materials = {{tag, 'modifiedGMPsteel', E, rho, fy, b_ratio, R0, cR1, cR2, a1, a2, a3, a4, sigma_init} }
    materials = { {1,'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 0.0, 55, 0, 55, 0} }
end
if (MatType == 2) then
    materials = { {1,'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 10, 60, 10, 60, 0} }
end
-- *******************************************************
model = StructureModel(2,2)
model:addNodes(nodes)
model:addMaterials(materials)
model:addElements(elements)
```

```
model:constrainNode(1,1,1)
model:constrainNode(2,1,1)
-- ******************************************************
function generateincrementalload()
 -- format:                    tag                   node dof
 local loadform = {'incrementalnodaldisplacement', 2  ,  1}
 local f = assert(io.open('cyclicwave.txt','r'))
 local n = f:read("*number")
 while (n ~= nil) do
  table.insert(loadform, 0.005*n)
  n = f:read("*number")
 end
 f:close()
 return loadform
end
 -- ******************************************************
staticloading = LoadDescription()
-- format: 'staticnodalload' <node> <dof> <amplitude>
l = generateincrementalload()
staticloading:addLoad(l)
-- ******************************************************
-- Static analysis
print("Static analysis started\n")
displ = {}
function displperstep(increment)
    dx2,dy2 = model:nodeDisplacements(2)
    table.insert(displ, dx2)
end
--
react = {}
function reactperstep(increment)
    fx1,fy1 = model:nodeRestoringForces(1)
    table.insert(react, fx1)
end
--
solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
analysis = StaticAnalysis(solver)
analysis:setStructureModel(model)
analysis:addcallback(displperstep,"increment")
analysis:addcallback(reactperstep,"increment")
analysis:solve(staticloading)
-- ******************************************************
-- Set output file
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
--
if (MatType == 1) then
    writedata1(displ,'Ex18StaticNonIsotropicNodalDisp_2.dat')
    writedata1(react,'Ex18StaticNonIsotropicReact_1.dat')
end
if (MatType == 2) then
    writedata1(displ,'Ex19StaticIsotropicNodalDisp_2.dat')
    writedata1(react,'Ex19StaticIsotropicReact_1.dat')
end
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

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

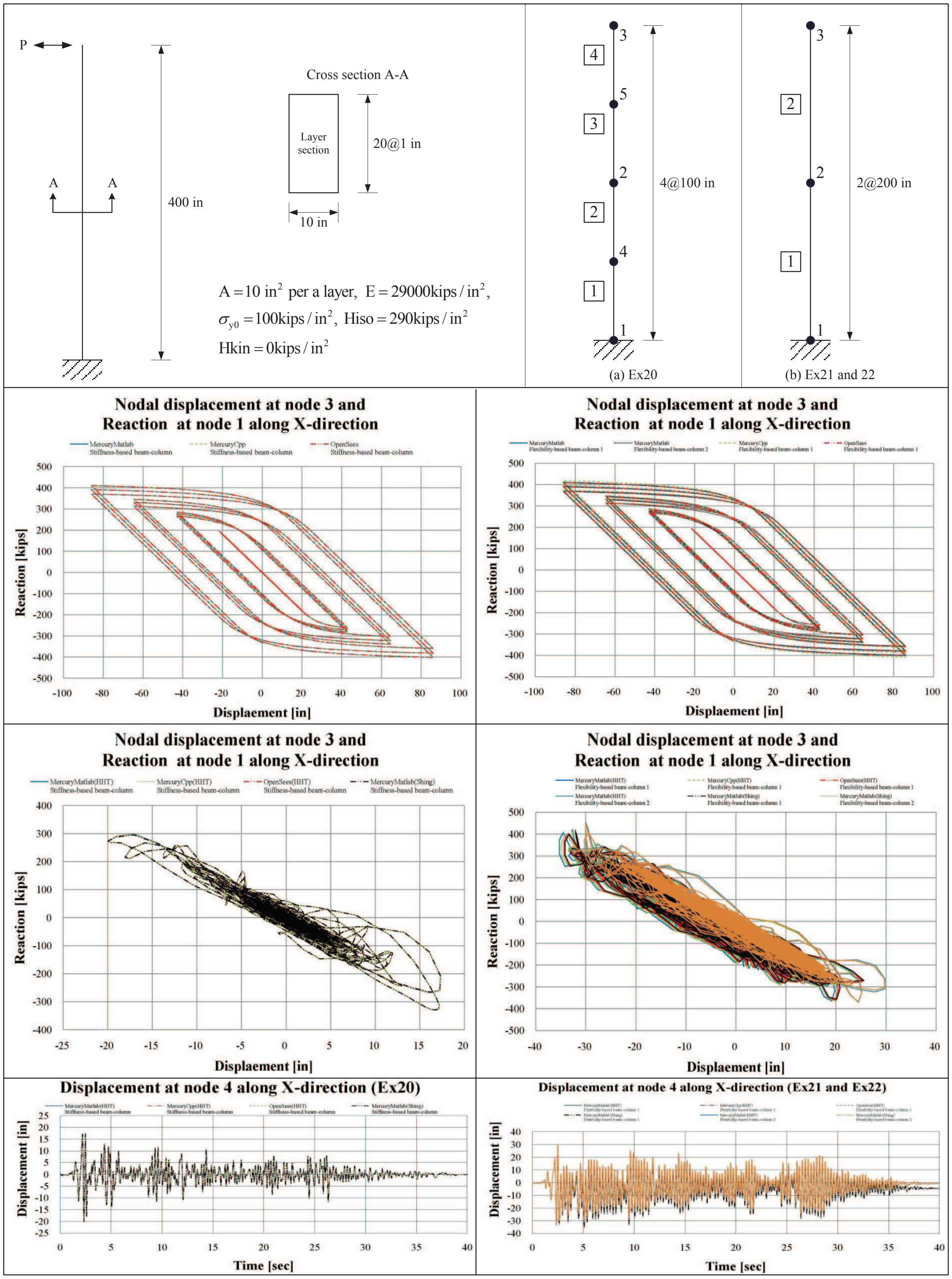


Figure 5: Examples 20- 22

```
% AnalysisType = 2: Transient analysis
AnalysisType = 2;
%----------------------------------------------------------
% Section DynType
% DynType = 1: Newmark beta
% DynType = 2: HHT
% DynType = 3: Shing
% DynType = 4: None
DynType = 3;
%----------------------------------------------------------
% Section EleType
% EleType = 1: Stiffness-based 2D beam-column (Ex20)
% EleType = 2: Flexibility-based 2D beam-column by Spacone (Ex21)
% EleType = 3: Flexibility-based 2D beam-column by Carol (Ex22)
EleType = 2;
%----------------------------------------------------------
% Preface
Unit      = {'kip', 'in'};
StrMode   = {2, 3};
%----------------------------------------------------------
% Control block
if ( (EleType == 1 || EleType == 2) && (DynType == 1 || DynType == 2) )
    Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
                            {'ModifiedNewtonRaphson', 1000, 1.0e-6, 'ForceNorm'};
                            {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};
                          };
                 'transient',{ {'NewtonRaphson', 10, 1.0e-3, 'DisplNorm'};
                               {'ModifiedNewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
                               {'InitialStiffness', 1000, 1.0e-3, 'DisplNorm'};
                          };
                 'element', { {'NewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
                              {'ModifiedNewtonRaphson', 1000, 1.0e-3, 'DisplNorm'};
                              {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'}
                            }
                 };
elseif ( (EleType == 3)  && (DynType == 1 || DynType == 2) )
    Iteration = {'static',{ {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};};
                 'transient',{ {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'};
                          };};
end
if (DynType == 3)
    Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
                            {'ModifiedNewtonRaphson', 1000, 1.0e-6, 'ForceNorm'};
                            {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};
                          };
                 'transient',{ {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'}; % for shing method
                             };
                 'element', { {'NewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
                              {'ModifiedNewtonRaphson', 1000, 1.0e-3, 'DisplNorm'};
                              {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'}
                            }
                 };
end


if (DynType == 1)
    Integration = {'Newmark', 0, 1/4, 1/2, 0, 0};
    eigens      = {0.02, 0.02};
end
if (DynType == 2)
    Integration = {'HHT', 0, -0.1, 0.3025, 0.6, 0, 0};
    eigens      = {0.02, 0.02};
end
if (DynType == 3)
    Integration = {'Shing', 0, -0.1, 0.3025, 0.6, 0, 0, 10};
    eigens      = {0.02, 0.02};
end
%----------------------------------------------------------
% Geometry block
if (EleType == 1)
    nodcoord = {1,   0, 0;
                4,   0, 100;
                2,   0, 200;
                5,   0, 300;
                3,   0, 400};
else
    nodcoord = {1,   0, 0;
                2,   0, 200;
                3,   0, 400};
end
%          nodtag, x, y, z
constraint = {1, 1, 1, 1};
%----------------------------------------------------------
% Element block
% elements = { { eletag, 'eletype', in, jn, nIp, sectag } };
if (EleType == 1)
    elements = { {1, 'StiffnessBased2DBeamColumn', 1, 4, 5, 1};
                 {2, 'StiffnessBased2DBeamColumn', 4, 2, 5, 1};
                 {3, 'StiffnessBased2DBeamColumn', 2, 5, 5, 1};
                 {4, 'StiffnessBased2DBeamColumn', 5, 3, 5, 1};};
elseif (EleType == 2)
    elements = { {1, 'FlexibilityBased2DBeamColumn1', 1, 2, 5, 1};
                 {2, 'FlexibilityBased2DBeamColumn1', 2, 3, 5, 1};};
elseif (EleType == 3)
    elements = { {1, 'FlexibilityBased2DBeamColumn2', 1, 2, 5, 1};
                 {2, 'FlexibilityBased2DBeamColumn2', 2, 3, 5, 1};};
end
%----------------------------------------------------------
% Section block
% b = 10, h = 20, number of layer = 10, hlayer = 2
area = 10; mtag  = 1; count = 0;
for ydis = -9.5:1.0:9.5
    count = count + 1; lay(count,1:3) = [mtag, area, ydis];
end
nlay = size(lay,1);
for i = 1:nlay
    laycell{i,1}=lay(i,1); laycell{i,2}=lay(i,2); laycell{i,3}=lay(i,3);
```

```
end
% sections = { sectag, 'Layer', {mattag, A, y} }
sections = {1, 'Layer', laycell };
clear area;clear mtag;clear count;clear nlay;clear lay;
clear laycell;clear i;clear ydis;
%------------------------------------------------------------
% Material block
% mass density = 15.2 (slug/ft^3)
%              = 15.2 (lb*s^2/ft/ft^3)
%              = 15.2*(10^-3)/(12^4) (kips*s^2/in^4)
materials = { {1, 'Hardening', 29*10^3, 100, 290, 0, 7.3302e-007};};
%------------------------------------------------------------
% Force block
if (AnalysisType == 1)
    DispInput = load('cyclicwave.txt');
    row = size(DispInput,1);
    for i = 1:row
        DispCell{i} = 50*DispInput(i);
    end
    forces = { 1, 'Static', {'NodalForces', {3, 1, 0} };
               2, 'DispCtrl', {3, 1, DispCell } };
    clear DispInput; clear row; clear i; clear DispCell;
elseif (AnalysisType == 2)
    ga = load('ElCentro_g_0_01_Matlab.txt');
    nga = size(ga, 1);
    for i = 1:nga
        groundacceleration{i,1} = ga(i,1);
        groundacceleration{i,2} = ga(i,2);
        groundacceleration{i,3} = ga(i,3);
    end
    forces = { 1, 'Static', {'NodalForces', {3, 1, 0} };
               2, 'Acceleration', {50*386.4, groundacceleration} };
    clear ga; clear nga; clear i; clear groundacceleration;
end
%------------------------------------------------------------
```

5.2 C++

```
-- ********************************************************
-- AnalysisType = 1: Static analysis
-- AnalysisType= 2: Trnasient analysis
AnalysisType = 2
-- EleType = 1: Stiffness-based beam-column (Ex20)
-- EleType = 2: Flexibility-based beam-column (Ex21)
EleType = 1
-- ********************************************************
if (EleType == 1) then
    nodes = {{1, 0, 0};
             {4, 0, 100, 'mass', 0.0146604, 0.0146604, 0};
             {2, 0, 200, 'mass', 0.0146604, 0.0146604, 0};
             {5, 0, 300, 'mass', 0.0146604, 0.0146604, 0};
             {3, 0, 400, 'mass', 0.0073302, 0.0073302, 0}};
    elements = { { 1, 'StiffnessBased2DBeamColumn', 1, 4, {1, 5} };
                 { 2, 'StiffnessBased2DBeamColumn', 4, 2, {1, 5} };
                 { 3, 'StiffnessBased2DBeamColumn', 2, 5, {1, 5} };
                 { 4, 'StiffnessBased2DBeamColumn', 5, 3, {1, 5} } };
end
if (EleType == 2) then
    nodes = {{1, 0, 0};
               {2, 0, 200, 'mass', 0.0293208, 0.0293208, 0};
               {3, 0, 400, 'mass', 0.0146604, 0.0146604, 0}}
    flexparams = {100000, 1e-3}
    elements = { { 1, 'FlexibilityBased2DBeamColumn', 1, 2, {1 ,5}, flexparams };
                 { 2, 'FlexibilityBased2DBeamColumn', 2, 3, {1 ,5}, flexparams } }
end
--

sections = {
1, 'Fiber',
-- MatTag, Area, y-loc, z-loc
{   1, 10, -9.5, 0;
    1, 10, -8.5, 0;
    1, 10, -7.5, 0;
    1, 10, -6.5, 0;
    1, 10, -5.5, 0;
    1, 10, -4.5, 0;
    1, 10, -3.5, 0;
    1, 10, -2.5, 0;
    1, 10, -1.5, 0;
    1, 10, -0.5, 0;
    1, 10,  0.5, 0;
    1, 10,  1.5, 0;
    1, 10,  2.5, 0;
    1, 10,  3.5, 0;
    1, 10,  4.5, 0;
    1, 10,  5.5, 0;
    1, 10,  6.5, 0;
    1, 10,  7.5, 0;
    1, 10,  8.5, 0;
    1, 10,  9.5, 0; } };
--
materials = { {1,'hardening',29000, 0, 100, 290, 0} }
-- ********************************************************
model = StructureModel(2,3)
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(sections)
model:addElements(elements)
model:constrainNode(1,1,1,1)
if (AnalysisType == 1) then
    model:constrainNode(3,1,0,0)
end
-- ********************************************************
```

```
-- Static analysis
if (AnalysisType == 1) then
    print("Static analysis started\n")
    -- ******************************************************
    function generateincrementalload()
     -- format:                  tag                        node dof
     local loadform = {'incrementalnodaldisplacement', 3  ,  1}
     local f = assert(io.open('cyclicwave.txt','r'))
     local n = f:read("*number")
     while (n ~= nil) do
      table.insert(loadform , 50*n)
      n = f:read("*number")
     end
     f:close()
     return loadform
    end
    -- ******************************************************
    staticloading = LoadDescription()
    -- format: 'staticnodalload' <node> <dof> <amplitude>
    l = generateincrementalload()
    staticloading:addLoad(l)
    -- ******************************************************
    displ = {}
    function displperstep(increment)
      dx3,dy3,dz3 = model:nodeDisplacements(3)
        table.insert(displ, dx3)
    end
    --
    react = {}
    function reactperstep(increment)
        fx1,fy1,fz1 = model:nodeRestoringForces(1)
        table.insert(react, fx1)
    end
    --
    --solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
    solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=1000000})
    analysis = StaticAnalysis(solver)
    analysis:setStructureModel(model)
    analysis:addcallback(displperstep,"increment")
    analysis:addcallback(reactperstep,"increment")
    analysis:solve(staticloading)
    -- *******************************************************
    -- Set output file
    function writedata1(x, fname)
        local f = assert(io.open(fname,'w'))
        local writenl = 0
        for i,v in ipairs(x) do
            f:write(v, " ")
            writenl = writenl + 1
            -- length of row size: writenl
            if (writenl > 0) then
                writenl = 0
                f:write("\n")
            end
        end
        f:close()
    end
    --
    if (EleType == 1) then
        writedata1(displ,'Ex20StaticNodalDisp_3.dat')
        writedata1(react,'Ex20StaticReact_1.dat')
    end
    if (EleType == 2) then
        writedata1(displ,'Ex21StaticNodalDisp_3.dat')
        writedata1(react,'Ex21StaticReact_1.dat')
    end
    print("Static analysis ended\n")
end
-- ********************************************************
if (AnalysisType == 2) then
    print("Transient analysis started\n")
    earthquakeloading = LoadDescription()
    accelamp = 50*386.4
    earthquakeloading:addLoad({'groundmotion','ElCentro_g_0_01_OpenSees.txt,dt=0.01', 1,accelamp})
    -- ******************************************************
    displ = {}
    function displpertime(time)
        dx3,dy3,dz3 = model:nodeDisplacements(3)
        table.insert(displ, dx3)
    end
  --
    react = {}
    function reactpertime(time)
        fx1,fy1,fz1 = model:nodeRestoringForces(1)
        table.insert(react, fx1)
    end
    -- ******************************************************
    -- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
    solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=1000})
  transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.1, 0.3025, 0.6)
    transientanalysis:addcallback(displpertime, "timestep")
    transientanalysis:addcallback(reactpertime, "timestep")
    if (EleType == 1) then
        model:setRayleighCoefficients(9.693343,0.000038)
    end
    if (EleType == 2) then
        model:setRayleighCoefficients(0.757867,0.000287)
    end
    transientanalysis:solve(4000)
    -- ******************************************************
    function writedata1(x, fname)
        local f = assert(io.open(fname,'w'))
        local writenl = 0
        for i,v in ipairs(x) do
            f:write(v, " ")
            writenl = writenl + 1
            -- length of row size: writenl
```

```
            if (writenl > 0) then
                writenl = 0
                f:write("\n")
            end
        end
        f:close()
    end
    --
    if (EleType == 1) then
        writedata1(displ,'Ex20HHTNodalDisp_3.dat')
        writedata1(react,'Ex20HHTReact_1.dat')
    end
    if (EleType == 2) then
        writedata1(displ,'Ex21HHTNodalDisp_3.dat')
        writedata1(react,'Ex21HHTReact_1.dat')
    end
    print("Transient analysis ended\n")
end
-- ******************************************************
```

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.

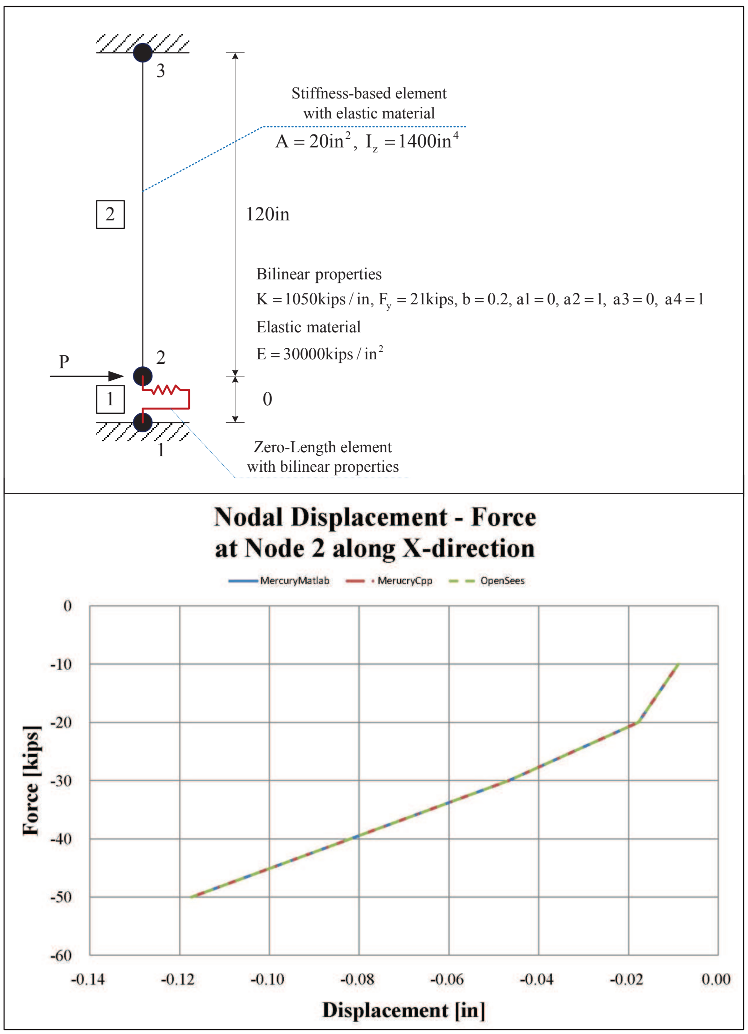


Figure 6: Examples 23

6.1 MATLAB

```
%==========================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written   : October 2009.
% File name: Ex23.m
%==========================================================
% Description
% 1. Static analysis
% 2. Static force and load control
% 3. Iterative method
% 4. Stiffness-based 2D beam column and Zero-Length 2D element
% 5. General section
% 6. Bilinear material
%----------------------------------------------------------
% Preface
Unit       = {'kip', 'in'};
StrMode    = {2, 3};
%----------------------------------------------------------
% Control block
Iteration = {'static',{ {'InitialStiffness', 100, 1.0e-8, 'ForceNorm'};
                       }
             };
%----------------------------------------------------------
% Geometry block
nodcoord    = {1,    0, 0;
               2,    0, 0;
               3,    0, 120};
constraint = {1, 1, 1, 1;
               3, 1, 1, 1};
%----------------------------------------------------------
% Element block
elements = { {1, 'ZeroLength2D', 1, 2, 0, 1, 0, deg2rad(90)}
             {2, 'StiffnessBased2DBeamColumn', 2, 3, 1, 1};};
%----------------------------------------------------------
% Section block
sections = { 1, 'General', {2, 20, 0, 0, 1400} };
%----------------------------------------------------------
% Material block
materials =  { {1, 'Bilinear', 1050, 21, 0.2, 0, 0, 1, 0, 1};
               {2, 'Elastic', 30000, 0, 0} };
%----------------------------------------------------------
% Force block
forces = { 1, 'Static',    {'NodalForces', {2, 1, -10}};
           2, 'LoadCtrl', {2, 1, {-20,-30,-40,-50} } };
%----------------------------------------------------------
```

6.2 C++

```
-- *********************************************************
nodes = { {1, 0, 0};
          {2, 0, 0};
          {3, 0, 120} };
-- *********************************************************
-- { eleTag, 'InterfaceElement2D', inode, jnode, { {matTag, {1,0,0} } }, { {secTag}}, {0,1,0},{-1,0,0} }
elements = { {1, 'InterfaceElement2D', 1, 2, { { 1, {1,0,0} } }, {} };
             {2, 'StiffnessBased2DBeamColumn', 2, 3, {1, 1} } };
-- *********************************************************
sections  = { 1, 'general', {2, 20, 1400} };
-- *********************************************************
materials = { {1,'bilinear', 1050, 0, 21, 0.2, 0, 1, 0, 1};
              {2,'elastic',30000, 0.0} };
-- *********************************************************
model = StructureModel(2,3)
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(sections)
model:addElements(elements)
-- *********************************************************
model:constrainNode(1,1,1,1)
model:constrainNode(3,1,1,1)
-- *********************************************************
staticloading = LoadDescription()
staticloading:addLoad({'incrementalnodalload', 2, 1, -10,-20,-30,-40,-50})
    -- *********************************************************
displ = {}
function displperstep(increment)
    dx2,dy2,dz2 = model:nodeDisplacements(2)
    table.insert(displ, dx2)
end
solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
analysis = StaticAnalysis(solver)
analysis:setStructureModel(model)
analysis:addcallback(displperstep,"increment")
analysis:solve(staticloading)
-- *********************************************************
-- Set output file
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
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.

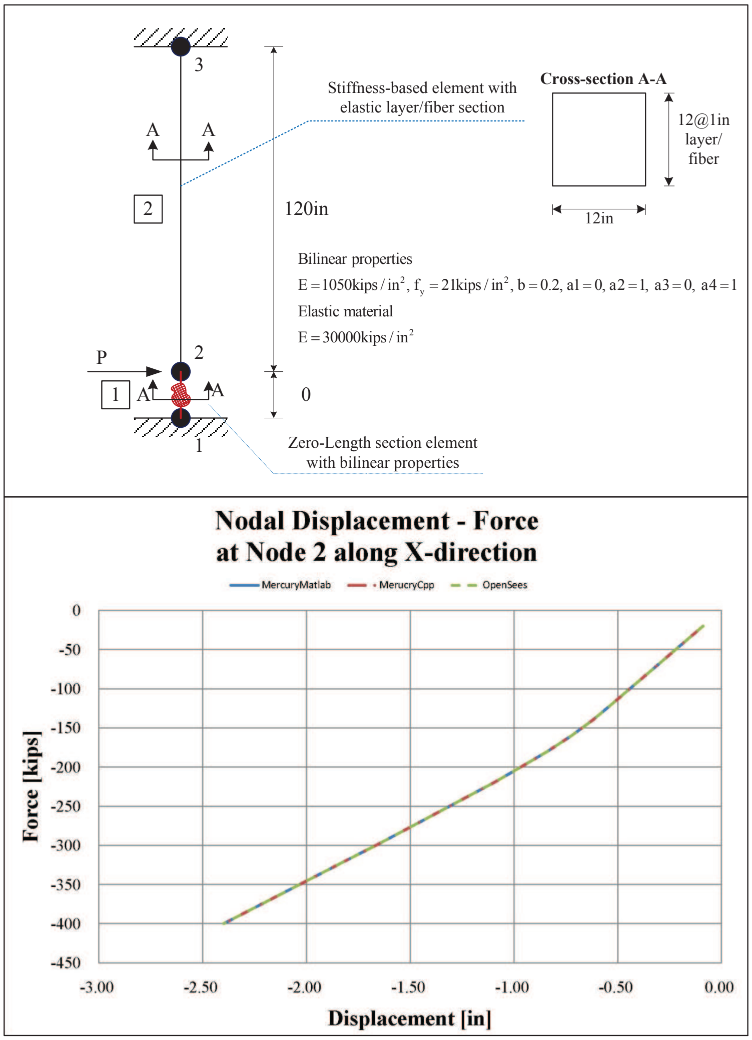


Figure 7: Examples 24

```
\subsection{MATLAB}
%==============================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written   : October 2009.
% File name: Ex24.m
%==============================================================
% Description
% 1. Static analysis
% 2. Static force and load control
% 3. Iterative method
% 4. Stiffness-based 2D beam column and Zero-Length 2D section element
% 5. General section
% 6. Bilinear material
%--------------------------------------------------------------
% Preface
Unit       = {'kip', 'in'};
StrMode    = {2, 3};
%--------------------------------------------------------------
% Control block
Iteration = {'static',{ {'InitialStiffness', 100, 1.0e-8, 'ForceNorm'};
                      }
            };
%--------------------------------------------------------------
% Geometry block
nodcoord    = {1,    0, 0;
```

```
                2,    0,  0;
                3,    0,  120};
constraint = {1,  1,  1,  1;
                3,  1,  1,  1};
%---------------------------------------------------------------------
% Element block
elements = { {1, 'ZeroLength2DSection', 1, 2, deg2rad(90), 1}
             {2, 'StiffnessBased2DBeamColumn', 2, 3, 3, 2};};
%---------------------------------------------------------------------
% Section block
sections = { 2, 'Layer', {2, 12, 5.5;
                          2, 12, 4.5;
                          2, 12, 3.5;
                          2, 12, 2.5;
                          2, 12, 1.5;
                          2, 12, 0.5;
                          2, 12, -0.5;
                          2, 12, -1.5;
                          2, 12, -2.5;
                          2, 12, -3.5;
                          2, 12, -4.5;
                          2, 12, -5.5};
             1, 'Layer', {1, 12, 5.5;
                          1, 12, 4.5;
                          1, 12, 3.5;
                          1, 12, 2.5;
                          1, 12, 1.5;
                          1, 12, 0.5;
                          1, 12, -0.5;
                          1, 12, -1.5;
                          1, 12, -2.5;
                          1, 12, -3.5;
                          1, 12, -4.5;
                          1, 12, -5.5} };
%---------------------------------------------------------------------
% Material block
materials =  { {1, 'Bilinear', 1050, 21, 0.2, 0, 0, 1, 0, 1};
               {2, 'Elastic', 30000, 0, 0} };
%---------------------------------------------------------------------
% Force block
forces = { 1, 'Static',    {'NodalForces', {2, 1, -20}};
           2, 'LoadCtrl', {2, 1, {-40, -60, -80, -100, -120,...
                                  -140, -160, -180, -200, -220,...
                                  -240, -260, -280, -300, -320,...
                                  -340, -360, -380, -400} } };
%---------------------------------------------------------------------------
```

7.1 C++

```
-- *********************************************************
nodes = { {1, 0, 0};
          {2, 0, 0};
          {3, 0, 120} };
-- *********************************************************
-- { eleTag, 'InterfaceElement2D', inode, jnode, { {matTag, {1,0,0} } }, { {secTag}}, {0,1,0},{-1,0,0} }
elements = { {1, 'InterfaceElement2D', 1, 2, {}, {{1}}, {0,1,0},{-1,0,0} };
             {2, 'StiffnessBased2DBeamColumn', 2, 3, {2, 3} } };
-- *********************************************************
sections = {1, 'Fiber', {1, 12, 5.5, 0;
                         1, 12, 4.5, 0;
                         1, 12, 3.5, 0;
                         1, 12, 2.5, 0;
                         1, 12, 1.5, 0;
                         1, 12, 0.5, 0;
                         1, 12, -0.5, 0;
                         1, 12, -1.5, 0;
                         1, 12, -2.5, 0;
                         1, 12, -3.5, 0;
                         1, 12, -4.5, 0;
                         1, 12, -5.5, 0};
                        2, 'Fiber', {2, 12, 5.5, 0;
                          2, 12, 4.5, 0;
                          2, 12, 3.5, 0;
                          2, 12, 2.5, 0;
                          2, 12, 1.5, 0;
                          2, 12, 0.5, 0;
                          2, 12, -0.5, 0;
                          2, 12, -1.5, 0;
                          2, 12, -2.5, 0;
                          2, 12, -3.5, 0;
                          2, 12, -4.5, 0;
                          2, 12, -5.5, 0};
              };
-- *********************************************************
materials = { {1,'bilinear', 1050, 0, 21, 0.2, 0, 1, 0, 1};
              {2,'elastic',30000, 0.0} };
-- *********************************************************
model = StructureModel(2,3)
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(sections)
model:addElements(elements)
-- *********************************************************
model:constrainNode(1,1,1,1)
model:constrainNode(3,1,1,1)
-- *********************************************************
staticloading = LoadDescription()
staticloading:addLoad({'incrementalnodalload', 2, 1, -20,-40,-60,-80,-100,-120,-140,-160,-180,-200,-220,-240,-260,-280,-300,-320,-340
    -- *********************************************************
displ = {}
function displperstep(increment)
    dx2,dy2,dz2 = model:nodeDisplacements(2)
    table.insert(displ, dx2)
```

```
end
solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
analysis = StaticAnalysis(solver)
analysis:setStructureModel(model)
analysis:addcallback(displperstep,"increment")
analysis:solve(staticloading)
-- ***************************************************
-- Set output file
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
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 facotr of -30 and 50 respectively. Ex25 has stiffness-based beam-column and Ex26 has flexibility-based beam-column.

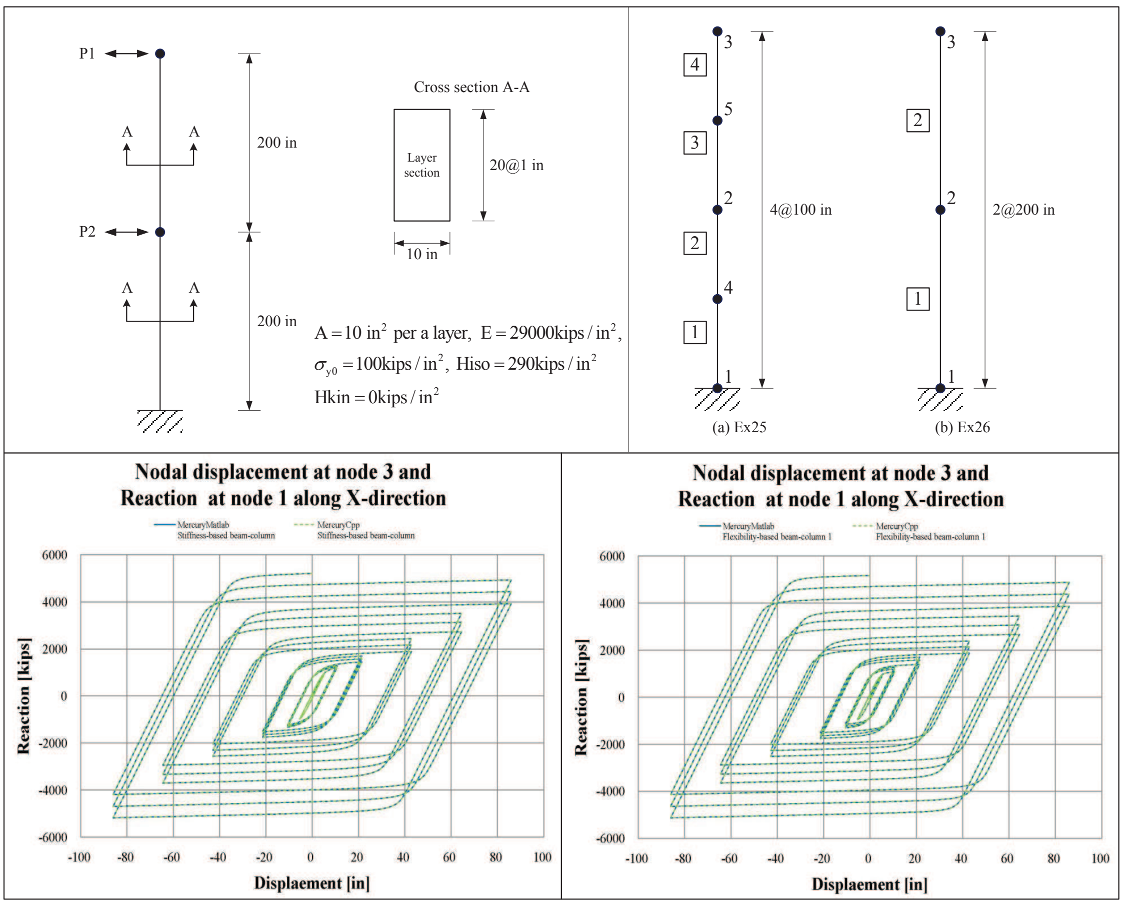


Figure 8: Examples 25- 26

8.1 MATLAB

```
%==========================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written   : October 2009.
% File name: Ex25to26.m
%==========================================================
% Description
% 1. Static analysis
% 2. Multiple displacement control
% 3. Iterative method
% 4. Stiffness-based 2D beam-column,
%    flexibility-based 2D beam-column 1 and 2
% 5. Layer sections
% 6. Hardening material
%----------------------------------------------------------
% Section EleType
% EleType = 1: Stiffness-based 2D beam-column (Ex25)
% EleType = 2: Flexibility-based 2D beam-column by Spacone (Ex26)
EleType = 2;
%----------------------------------------------------------
% Preface
Unit      = {'kip', 'in'};
StrMode   = {2, 3};
%----------------------------------------------------------
% Control block
Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-8, 'ForceNorm'};
                        {'ModifiedNewtonRaphson', 1000, 1.0e-8, 'ForceNorm'};
                        {'InitialStiffness', 10000, 1.0e-8, 'ForceNorm'};
                      };
             'element', { {'NewtonRaphson', 100, 1.0e-5, 'DisplNorm'};
                          {'ModifiedNewtonRaphson', 1000, 1.0e-5, 'DisplNorm'};
                          {'InitialStiffness', 100000, 1.0e-5, 'DisplNorm'}
                        };
            };
%----------------------------------------------------------
% Geometry block
if (EleType == 1)
    nodcoord = {1,   0, 0;
                4,   0, 100;
                2,   0, 200;
                5,   0, 300;
                3,   0, 400};
elseif (EleType == 2)
    nodcoord = {1,   0, 0;
                2,   0, 200;
                3,   0, 400};
end
%          nodtag, x, y, z
constraint = {1, 1, 1, 1};
%----------------------------------------------------------
% Element block
% elements = { { eletag, 'eletype', in, jn, nIp, sectag } };
if (EleType == 1)
    elements = { {1, 'StiffnessBased2DBeamColumn', 1, 4, 5, 1};
                 {2, 'StiffnessBased2DBeamColumn', 4, 2, 5, 1};
                 {3, 'StiffnessBased2DBeamColumn', 2, 5, 5, 1};
                 {4, 'StiffnessBased2DBeamColumn', 5, 3, 5, 1};};
elseif (EleType == 2)
    elements = { {1, 'FlexibilityBased2DBeamColumn1', 1, 2, 5, 1};
                 {2, 'FlexibilityBased2DBeamColumn1', 2, 3, 5, 1};};
end
%----------------------------------------------------------
% Section block
% b = 10, h = 20, number of layer = 10, hlayer = 2
area = 10; mtag  = 1; count = 0;
for ydis =  -9.5:1.0:9.5
    count = count + 1; lay(count,1:3) = [mtag, area, ydis];
end
nlay = size(lay,1);
for i = 1:nlay
    laycell{i,1}=lay(i,1); laycell{i,2}=lay(i,2); laycell{i,3}=lay(i,3);
end
% sections = { sectag, 'Layer', {mattag, A, y} }
sections =  {1, 'Layer', laycell };
clear area;clear mtag;clear count;clear nlay;clear lay;
clear laycell;clear i;clear ydis;
%----------------------------------------------------------
% Material block
% mass density = 15.2 (slug/ft^3)
%              = 15.2 (lb*s^2/ft/ft^3)
%              = 15.2*(10^-3)/(12^4) (kips*s^2/in^4)
materials = { {1, 'Hardening', 29*10^3, 100, 290, 0, 7.3302e-007};};
%----------------------------------------------------------
% Force block
DispInput = load('cyclicwave.txt');
row = size(DispInput,1);
for i = 1:row
    DispCell1{i} = -30*DispInput(i);
    DispCell2{i} = 50*DispInput(i);
end
forces = { 1, 'Static', {'NodalForces', {3, 1, 0} };
           2, 'DispCtrl', {2, 1, DispCell1;
                           3, 1, DispCell2 } };
clear DispInput; clear row; clear i; clear DispCell1;clear DispCell2;
%----------------------------------------------------------
```

8.2 C++

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

```
EleType = 2
-- ********************************************************
if (EleType == 1) then
    nodes = {{1, 0, 0},
           {4, 0, 100},
           {2, 0, 200},
           {5, 0, 300},
           {3, 0, 400}};
    elements = { { 1, 'StiffnessBased2DBeamColumn', 1, 4, {1, 5} };
                 { 2, 'StiffnessBased2DBeamColumn', 4, 2, {1, 5} };
                 { 3, 'StiffnessBased2DBeamColumn', 2, 5, {1, 5} };
                 { 4, 'StiffnessBased2DBeamColumn', 5, 3, {1, 5} } };
end

if (EleType == 2) then
    nodes = {{1, 0, 0},
             {2, 0, 200},
             {3, 0, 400}}
    flexparams = {1000, 1e-5}
    elements = { { 1, 'FlexibilityBased2DBeamColumn', 1, 2, {1 ,5}, flexparams };
                 { 2, 'FlexibilityBased2DBeamColumn', 2, 3, {1 ,5}, flexparams } }
end
-- ********************************************************
sections = {
1, 'Fiber',
-- MatTag, Area, y-loc, z-loc
{   1, 10, -9.5, 0;
    1, 10, -8.5, 0;
    1, 10, -7.5, 0;
    1, 10, -6.5, 0;
    1, 10, -5.5, 0;
    1, 10, -4.5, 0;
    1, 10, -3.5, 0;
    1, 10, -2.5, 0;
    1, 10, -1.5, 0;
    1, 10, -0.5, 0;
    1, 10,  0.5, 0;
    1, 10,  1.5, 0;
    1, 10,  2.5, 0;
    1, 10,  3.5, 0;
    1, 10,  4.5, 0;
    1, 10,  5.5, 0;
    1, 10,  6.5, 0;
    1, 10,  7.5, 0;
    1, 10,  8.5, 0;
    1, 10,  9.5, 0; } };
-- ********************************************************
materials = { {1,'hardening',29000, 0, 100, 290, 0} }
-- ********************************************************
model = StructureModel(2,3)
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(sections)
model:addElements(elements)
model:constrainNode(1,1,1,1)
model:constrainNode(2,1,0,0)
model:constrainNode(3,1,0,0)
-- ********************************************************
-- Static analysis
function generateincrementalload2()
 -- format:                tag                  node dof
    local loadform = {'incrementalnodaldisplacement', 2  ,  1}
    local f = assert(io.open('cyclicwave.txt','r'))
    local n = f:read("*number")
    while (n ~= nil) do
     table.insert(loadform, -30*n)
     n = f:read("*number")
    end
    f:close()
    return loadform
end
function generateincrementalload3()
 -- format:                tag                  node dof
    local loadform = {'incrementalnodaldisplacement', 3  ,  1}
    local f = assert(io.open('cyclicwave.txt','r'))
    local n = f:read("*number")
    while (n ~= nil) do
     table.insert(loadform, 50*n)
     n = f:read("*number")
    end
    f:close()
    return loadform
end
    -- ********************************************************
staticloading = LoadDescription()
-- format: 'staticnodalload' <node> <dof> <amplitude>
l2 = generateincrementalload2()
l3 = generateincrementalload3()
staticloading:addLoad(l2)
staticloading:addLoad(l3)
-- ********************************************************
displ = {}
function displperstep(increment)
  dx2,dy2,dz2 = model:nodeDisplacements(2)
  dx3,dy3,dz3 = model:nodeDisplacements(3)
    table.insert(displ, dx2)
    table.insert(displ, dx3)
end
--
react = {}
function reactperstep(increment)
    fx1,fy1,fz1 = model:nodeRestoringForces(1)
    print("Work\n");
    table.insert(react, fx1)
end
-- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
-- solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-5, iterations=10000})
```

```
--              multisolver <type> <disp delta> <residual> <max iterations>
solver = NonlinearSolver("multisolver",
                                              "newtonraphson", 1e-5, 1e-5, 100,
                                              "initialstiffness", 1e-5, 1e-5, 1000)



analysis = StaticAnalysis(solver)
analysis:setStructureModel(model)
analysis:addcallback(displperstep,"increment")
analysis:addcallback(reactperstep,"increment")
analysis:solve(staticloading)
-- ***********************************************************
-- Set output file
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
function writedata2(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 1) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
--
if (EleType == 1) then
    writedata2(displ,'Ex25NodalDisp_2_3.dat')
    writedata1(react,'Ex25React_1.dat')
end
if (EleType == 2) then
    writedata2(displ,'Ex26NodalDisp_2_3.dat')
    writedata1(react,'Ex26React_1.dat')
end
```

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

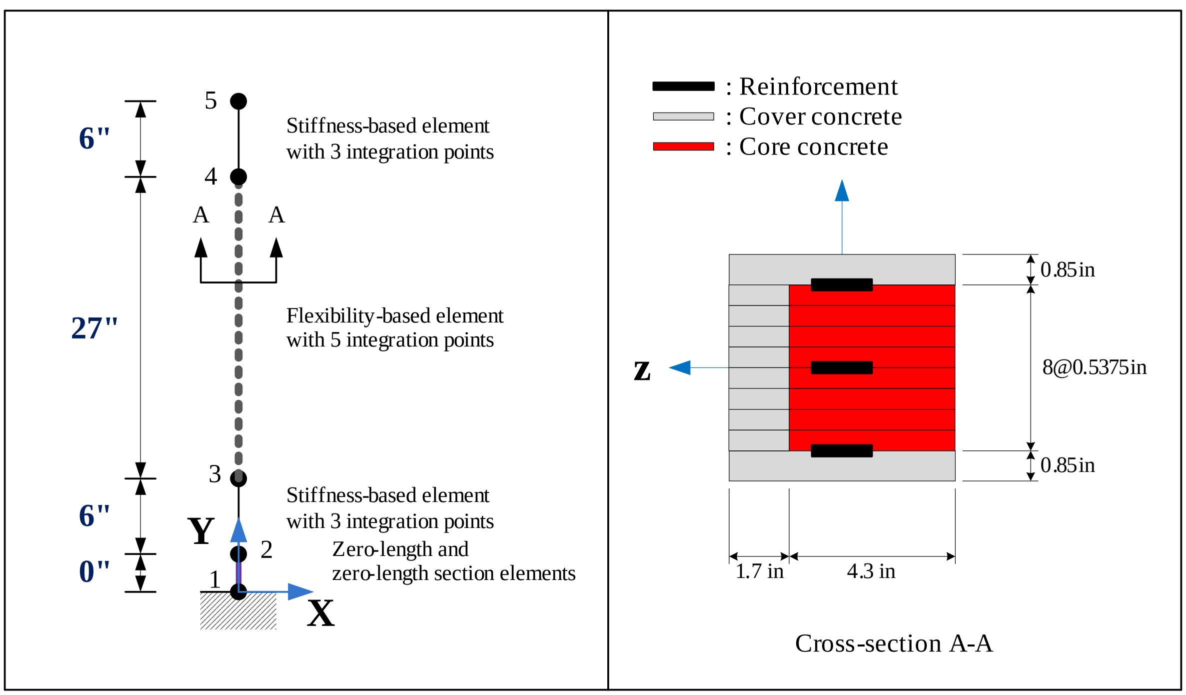


Figure 9: Beam-column elements for Ex27 and Ex28

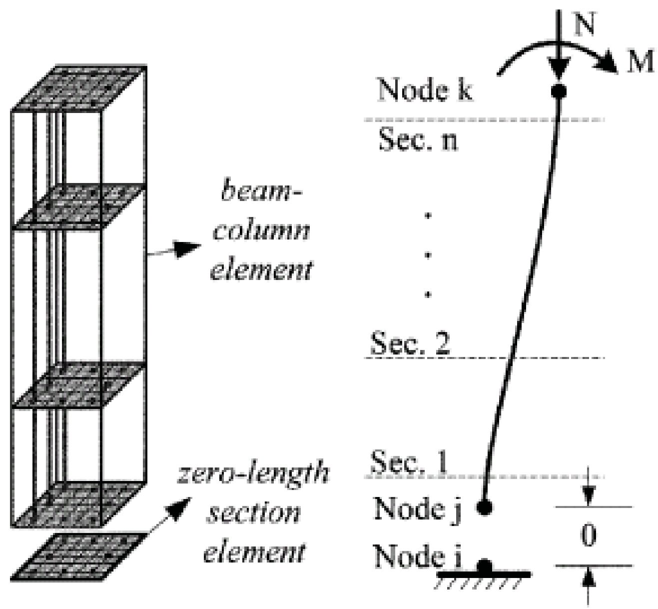


Figure 10: Bar slip zero-length fiber section element (?)

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

```
%=================================================================
% 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

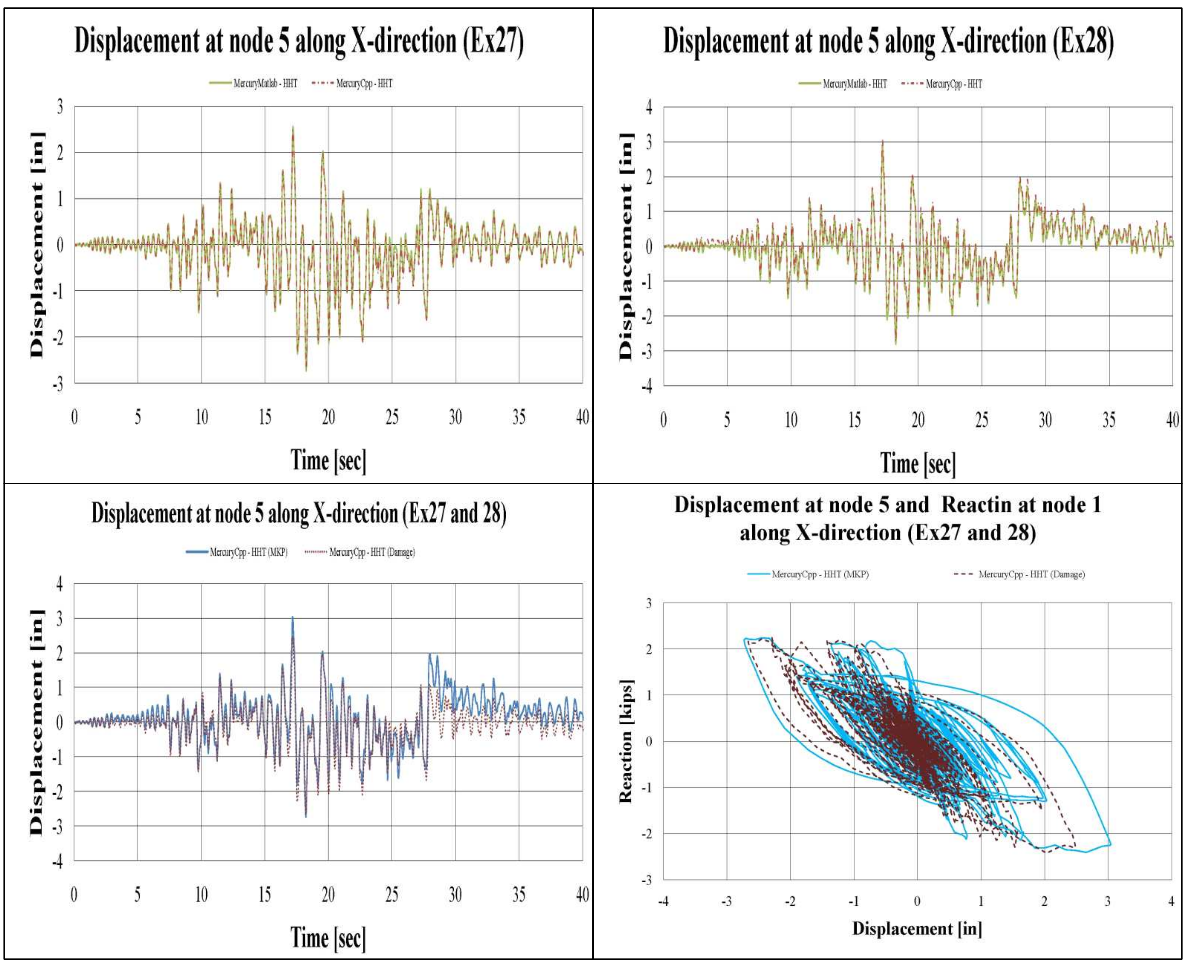


Figure 11: Output for Ex27 and Ex28

```
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written   : October 2009.
% File name: Ex27.m (1 column)
%==========================================================================
% Preface
Unit       = {'kip', 'in'};
%            ndim, ndofpn
StrMode    = {2, 3};
%--------------------------------------------------------------------------
% Control block
Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
                        {'InitialStiffness', 10000, 1.0e-6, 'DisplNorm'};
                      }
             'element',{ {'NewtonRaphson', 1000, 1.0e-6, 'DisplNorm'};
                         {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
                       }
             'transient',{ {'NewtonRaphson', 100, 1.0e-6, 'DisplNorm'};
                           {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
                         }
            };
%Integration = {'HHT', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252};
Integration = {'Shing', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252,10};
addMass     = {1, 0, 0, 0;
               2, 2.50712E-05, 2.50712E-05, 0;
               3, 0.000137891, 0.000137891, 0;
               4, 0.000137891, 0.000137891, 0;
               5, 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0};
%--------------------------------------------------------------------------
nodcoord = {1, 0,  0;
            2, 0,  0;
            3, 0,  6;
            4, 0, 33;
            5, 0, 39};
constraint = {1, 1, 1, 1};
%--------------------------------------------------------------------------

elements = { {1, 'ZeroLength2D', 1, 2, 0, 26, 0, pi()/2};
             {2, 'ZeroLength2DSection', 1, 2, pi()/2, 3};
             {3, 'StiffnessBased2DBeamColumn', 2, 3, 3, 1};
             {4, 'FlexibilityBased2DBeamColumn1', 3, 4, 5, 1};
             {5, 'StiffnessBased2DBeamColumn', 4, 5, 3, 1}};
%--------------------------------------------------------------------------
sections = {1, 'Layer', {1, 5.1,     2.575;
                         1, 5.1,    -2.575;
                         1, 0.91375,    1.88125;
                         1, 0.91375,    1.34375;
                         1, 0.91375,    0.80625;
                         1, 0.91375,    0.26875;
                         1, 0.91375,   -0.26875;
                         1, 0.91375,   -0.80625;
                         1, 0.91375,   -1.34375;
                         1, 0.91375,   -1.88125;
                         2, 2.31125,    1.88125;
                         2, 2.31125,    1.34375;
                         2, 2.31125,    0.80625;
                         2, 2.31125,    0.26875;
                         2, 2.31125,   -0.26875;
                         2, 2.31125,   -0.80625;
                         2, 2.31125,   -1.34375;
                         2, 2.31125,   -1.88125;
                        17, 0.147,  2.15;
                        17, 0.098,  0;
                        17, 0.147, -2.15;};
            3, 'Layer', {5, 5.1,     2.575;
                         5, 5.1,    -2.575;
                         5, 0.91375,    1.88125;
                         5, 0.91375,    1.34375;
                         5, 0.91375,    0.80625;
                         5, 0.91375,    0.26875;
                         5, 0.91375,   -0.26875;
                         5, 0.91375,   -0.80625;
                         5, 0.91375,   -1.34375;
                         5, 0.91375,   -1.88125;
                         6, 2.31125,    1.88125;
                         6, 2.31125,    1.34375;
                         6, 2.31125,    0.80625;
                         6, 2.31125,    0.26875;
                         6, 2.31125,   -0.26875;
                         6, 2.31125,   -0.80625;
                         6, 2.31125,   -1.34375;
                         6, 2.31125,   -1.88125;
                        19, 0.147,  2.15;
                        19, 0.098,  0;
                        19, 0.147, -2.15;};};
%--------------------------------------------------------------------------
materials = {   {1, 'ModKP',-3.57,-0.0026,  -1.19    ,-0.0078, 0.3, 0.448121077, 549.2307692,0 };
                {2, 'ModKP',-7.5,-0.00546,-7.35,     -0.01638,0.3, 0.649519053,  549.4505495,0};
                {   5, 'ModKP',-3.57,-0.0136,-1.19,-0.0408,0.3,0.448121077,105,0};
                {   6, 'ModKP',-7.5,     -0.0288,-6.75,-0.0864,0.3,0.649519053,104.1666667,0};
                {   17, 'ModGMP',26500, 87.5,0.01,15,0.925,0.15,0,0,55,0,55};
                {   19, 'ModGMP',6949,87.5,0.01,15, 0.925,0.15,0,0,55,0,55};
                {   26, 'Bilinear',1690,78.2     ,0.173,0, 0,55,0,55};};
%--------------------------------------------------------------------------
% Force block
ga = load('NR_g_dt_0_01_Matlab.txt');
nga = size(ga, 1);
for i = 1:nga
    groundacceleration{i,1} = ga(i,1);
    groundacceleration{i,2} = ga(i,2);
    groundacceleration{i,3} = ga(i,3);
end
forces = { 1, 'Static', {'NodalForces', {5, 1, 0} };
           2, 'Acceleration', {386.4, groundacceleration} };
```

9.2 C++

```
-- ********************************************************
-- 1 column
-- ********************************************************
--    o    (39)
--    |         Stiffness-based beam-column with 2 integration points
--    o    (33)
--    |
--    |         Flexibility-based beam-column with 5 integration points
--    |
--    o    (6)
--    |         Stiffness-based beam-column with 2 integration points
--    o    (0)
--              Zero-Length and zero-Length section elements(bottom bar slip and shear deformation)
--   _o_   (0) (Fixed support)

-- ********************************************************
elements = {}
-- -----------
-- create ductile column node coordinates
-- create nodes
nodes = {{1, 0,  0, 'mass', 0, 0,0};
         {2, 0,  0, 'mass', 2.50712E-05, 2.50712E-05,0};
         {3, 0,  6, 'mass', 0.000137891, 0.000137891, 0};
         {4, 0, 33, 'mass', 0.000137891, 0.000137891, 0};
         {5, 0, 39, 'mass', 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0} }
-- figure out section names
barslipsectionf     = 'BSColDFSection'
barslipspringf      = 'BSColDFSS'
columnsection       = 'ColDSection'
columnrigidsection = 'ColRigidSection'
nIp_stif = 3;
nIp_flex = 5;
flexparams = {10000,1e-6}
-- Define elements
barslipbottom   = { 1, 'InterfaceElement2D', 1, 2, { {barslipspringf, {1,0,0} } },
                     { {barslipsectionf}}, {0,1,0},{-1,0,0} }
plasticcolumn1 = { 2, 'StiffnessBased2DBeamColumn', 2, 3, {columnsection, nIp_stif} }
flexcolumn      = { 3, 'FlexibilityBased2DBeamColumn', 3, 4, {columnsection,nIp_flex}, flexparams }
plasticcolumn2 = { 4, 'StiffnessBased2DBeamColumn',4, 5, {columnsection, nIp_stif} }
table.insert(elements, barslipbottom)
table.insert(elements, plasticcolumn1)
table.insert(elements, flexcolumn)
table.insert(elements, plasticcolumn2)
-- ********************************************************
-- Set section properties.
--
allsections = {
-- -----------
'ColDSection' , 'Fiber',
-- Tag, Area, y-loc, z-loc
{   'ColDCover' ,   5.1 ,     2.575   ,    0    ,
    'ColDCover' ,   5.1 ,    -2.575   ,    0    ,
    'ColDCover' ,   0.91375 ,   1.88125 ,   0   ,
    'ColDCover' ,   0.91375 ,   1.34375 ,   0   ,
    'ColDCover' ,   0.91375 ,   0.80625 ,   0   ,
    'ColDCover' ,   0.91375 ,   0.26875 ,   0   ,
    'ColDCover' ,   0.91375 ,   -0.26875    ,   0   ,
    'ColDCover' ,   0.91375 ,   -0.80625    ,   0   ,
    'ColDCover' ,   0.91375 ,   -1.34375    ,   0   ,
    'ColDCover' ,   0.91375 ,   -1.88125    ,   0   ,
    'ColDCore'  ,   2.31125 ,   1.88125 ,   0   ,
    'ColDCore'  ,   2.31125 ,   1.34375 ,   0   ,
    'ColDCore'  ,   2.31125 ,   0.80625 ,   0   ,
    'ColDCore'  ,   2.31125 ,   0.26875 ,   0   ,
    'ColDCore'  ,   2.31125 ,   -0.26875    ,   0   ,
    'ColDCore'  ,   2.31125 ,   -0.80625    ,   0   ,
    'ColDCore'  ,   2.31125 ,   -1.34375    ,   0   ,
    'ColDCore'  ,   2.31125 ,   -1.88125    ,   0   ,
    'ColDSteel' ,   0.147   ,   2.15    ,   0   ,
    'ColDSteel' ,   0.098   ,   0   ,   0   ,
    'ColDSteel' ,   0.147   ,   -2.15   ,   0   };
-- -----------
'BSColDFSection'   , 'Fiber',
{   'BSColDFCover'  ,   5.1 ,   2.575   ,   0   ,
    'BSColDFCover'  ,   5.1 ,   -2.575  ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   1.88125 ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   1.34375 ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   0.80625 ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   0.26875 ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   -0.26875    ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   -0.80625    ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   -1.34375    ,   0   ,
    'BSColDFCover'  ,   0.91375 ,   -1.88125    ,   0   ,
    'BSColDFCore'   ,   2.31125 ,   1.88125 ,   0   ,
    'BSColDFCore'   ,   2.31125 ,   1.34375 ,   0   ,
    'BSColDFCore'   ,   2.31125 ,   0.80625 ,   0   ,
    'BSColDFCore' , 2.31125 ,   0.26875 ,   0   ,
    'BSColDFCore'   ,   2.31125 ,   -0.26875    ,   0   ,
    'BSColDFCore'   ,   2.31125 ,   -0.80625    ,   0   ,
    'BSColDFCore'   ,   2.31125 ,   -1.34375    ,   0   ,
    'BSColDFCore'   ,   2.31125 ,   -1.88125    ,   0   ,
    'BSColDFSteel'  ,   0.147   ,   2.15    ,   0   ,
    'BSColDFSteel'  ,   0.098   ,   0   ,   0   ,
    'BSColDFSteel'  ,   0.147   ,   -2.15   ,   0   }; }
-- ********************************************************
-- Set material properties.
concretemat = 'ConcreteLinearTensionSoftening'
steelmat    = 'GiuffreMenegottoPinto'
sspringmat  = 'Bilinear'
materials = {
{'ColDCover'     , concretemat , 549.2307692 , 0, -3.57, -0.0026 , -1.19, -0.0078 , 0.3, 0.448121077},
{'ColDCore'      , concretemat , 549.4505495 , 0, -7.5 , -0.00546, -7.35, -0.01638, 0.3, 0.649519053},
{'BSColDFCover'  , concretemat , 105         , 0, -3.57, -0.0136 , -1.19, -0.0408 , 0.3, 0.448121077},
{'BSColDFCore'   , concretemat , 104.1666667 , 0, -7.5 , -0.0288 , -6.75, -0.0864 , 0.3, 0.649519053},
```

```
{'ColDSteel'      , steelmat , 26500, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSColDFSteel'  , steelmat , 6949  , 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSColDFSS'  , sspringmat, 1690,    0, 78.2, 0.173,  0, 55,  0, 55},
}
-- ***********************************************************************
function dumptable(x)
    local result = '{'
    for i,v in ipairs(x) do
        if (type(v) == 'table') then
            result = result .. dumptable(v)
        else
            result = result .. v .. ','
        end
    end
    return result .. '}\n'
end
-- ------------
--print(dumptable(nodes))
--print(dumptable(elements))
--print(dumptable(materials))
--print(dumptable(allsections))
-- ***********************************************************************
-- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
model = StructureModel(2,3)
-- Assign all input data to Mercury
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(allsections)
model:addElements(elements)
-- ***********************************************************************
-- constrain bottom nodes
model:constrainNode(1,1,1,1)
-- ***********************************************************************
print("Transient analysis started\n")
earthquakeloading = LoadDescription()
earthquakeloading:addLoad({'groundmotion','NR_g_dt_0_01_OpneSees.txt,dt=0.01', 1, 386.4})
-- *****************************************************
displ = {}
function displpertime(time)
    dx5,dy5,dz5 = model:nodeDisplacements(5)
    table.insert(displ, dx5)
    print("Work\n");
end
--
react = {}
function reactpertime(time)
    fx1,fy1 = model:nodeRestoringForces(1)
    table.insert(react, fx1)
end
-- *******************************************************
solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-6, iterations=10000})
transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.2, 0.36, 0.7)
transientanalysis:addcallback(displpertime, "timestep")
transientanalysis:addcallback(reactpertime, "timestep")
model:setRayleighCoefficients(0.6318799279194399, 0.00015503501814608252)
transientanalysis:solve(7641)
-- *****************************************************
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
writedata1(displ,'Ex27HHTNodalDisp_5.dat')
writedata1(react,'Ex27HHTReact_1.dat')
print("Transient analysis ended\n")
```

9.3 MATLAB

```
%==========================================================================
% Mercury Matlab Version 1.0.1
% Written by Dae-Hung Kang, CU-NEES
% Copyright 2009, CU-NEES
% Written   : October 2009.
% File name: Ex28.m (1 column)
%==========================================================================
% Preface
Unit      = {'kip', 'in'};
%           ndim, ndofpn
StrMode   = {2, 3};
%--------------------------------------------------------------------------
% Control block
Iteration = {'static',{ {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
                        {'InitialStiffness', 10000, 1.0e-6, 'DisplNorm'};
                      }
             'element',{ {'NewtonRaphson', 10, 1.0e-6, 'DisplNorm'};
                         {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
                       }
             'transient',{ {'NewtonRaphson', 10, 1.0e-6, 'DisplNorm'};
                           {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
                         }
            };
Integration = {'HHT', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252};
addMass     = {1, 0, 0, 0;
               2, 2.50712E-05, 2.50712E-05, 0;
```

```
                 3, 0.000137891, 0.000137891, 0;
                 4, 0.000137891, 0.000137891, 0;
                 5, 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0};
%---------------------------------------------------------------------------
nodcoord = {1, 0,  0;
             2, 0,  0;
             3, 0,  6;
             4, 0, 33;
             5, 0, 39};
constraint = {1, 1, 1, 1};
%---------------------------------------------------------------------------

elements = { {1, 'ZeroLength2D', 1, 2, 0, 26, 0, pi()/2};
             {2, 'ZeroLength2DSection', 1, 2, pi()/2, 3};
             {3, 'StiffnessBased2DBeamColumn', 2, 3, 3, 1};
             {4, 'FlexibilityBased2DBeamColumn1', 3, 4, 5, 1};
             {5, 'StiffnessBased2DBeamColumn', 4, 5, 3, 1}};
%---------------------------------------------------------------------------
sections = {1, 'Layer', {1, 5.1,     2.575;
                          1, 5.1,    -2.575;
                          1, 0.91375,     1.88125;
                          1, 0.91375,     1.34375;
                          1, 0.91375,     0.80625;
                          1, 0.91375,     0.26875;
                          1, 0.91375,    -0.26875;
                          1, 0.91375,    -0.80625;
                          1, 0.91375,    -1.34375;
                          1, 0.91375,    -1.88125;
                          2, 2.31125,     1.88125;
                          2, 2.31125,     1.34375;
                          2, 2.31125,     0.80625;
                          2, 2.31125,     0.26875;
                          2, 2.31125,    -0.26875;
                          2, 2.31125,    -0.80625;
                          2, 2.31125,    -1.34375;
                          2, 2.31125,    -1.88125;
                         17, 0.147,   2.15;
                         17, 0.098,   0;
                         17, 0.147,  -2.15;};
             3, 'Layer', {5, 5.1,     2.575;
                          5, 5.1,    -2.575;
                          5, 0.91375,     1.88125;
                          5, 0.91375,     1.34375;
                          5, 0.91375,     0.80625;
                          5, 0.91375,     0.26875;
                          5, 0.91375,    -0.26875;
                          5, 0.91375,    -0.80625;
                          5, 0.91375,    -1.34375;
                          5, 0.91375,    -1.88125;
                          6, 2.31125,     1.88125;
                          6, 2.31125,     1.34375;
                          6, 2.31125,     0.80625;
                          6, 2.31125,     0.26875;
                          6, 2.31125,    -0.26875;
                          6, 2.31125,    -0.80625;
                          6, 2.31125,    -1.34375;
                          6, 2.31125,    -1.88125;
                         19, 0.147,   2.15;
                         19, 0.098,   0;
                         19, 0.147,  -2.15;};};
%---------------------------------------------------------------------------
materials = {     {1,'AnisotropicDamage',2829,0.2,5.855E-08,1870,0.003248,0.05168,38890000000,22.27,0.9999999,0};
                {     2,'AnisotropicDamage',1804,0.2,0.000006483,503.6,3.125E-07,0.4054,3159000000,102,0.9999999,0};
                {    5,'AnisotropicDamage',569.3,0.2,1.689E-19,371.8,0.832,1.461,9.529E+13     ,100.4,0.9999999,0};
                {     6,'AnisotropicDamage',396.7,0.2,0.0008326,124.8,2.084E-07,1.61,8.856E+11,62.48,0.9999999,0};
                {    17,'ModGMP',26500,87.5,0.01,15,0.925,0.15,0,0,55,0,55};
                {    19,'ModGMP',6949,87.5,0.01,15,0.925,0.15,0,0,55,0,55     };
                {    26,'Bilinear',1690,78.2,     0.173,0,0,55,0,55};};
%---------------------------------------------------------------------------
% Force block
ga = load('NR_g_dt_0_01_Matlab.txt');
nga = size(ga, 1);
for i = 1:nga
    groundacceleration{i,1} = ga(i,1);
    groundacceleration{i,2} = ga(i,2);
    groundacceleration{i,3} = ga(i,3);
end
forces = { 1, 'Static', {'NodalForces', {5, 1, 0} };
           2, 'Acceleration', {386.4, groundacceleration} };
```

9.4 C++

```
-- ********************************************************
-- 1 column
-- ********************************************************
--    o   (39)
--    |         Stiffness-based beam-column with 2 integration points
--    o   (33)
--    |
--    |         Flexibility-based beam-column with 5 integration points
--    |
--    o   (6)
--    |         Stiffness-based beam-column with 2 integration points
--    o   (0)
--              Zero-Length and zero-Length section elements(bottom bar slip and shear deformation)
--   _o_  (0) (Fixed support)

-- ********************************************************
elements = {}
-- ------------
-- create ductile column node coordinates
-- create nodes
nodes = {{1, 0,  0, 'mass', 0, 0,0};
```

```
          {2, 0,  0, 'mass', 2.50712E-05, 2.50712E-05,0};
          {3, 0,  6, 'mass', 0.000137891, 0.000137891, 0};
          {4, 0, 33, 'mass', 0.000137891, 0.000137891, 0};
          {5, 0, 39, 'mass', 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0} }
-- figure out section names
barslipsectionf     = 'BSColDFSection'
barslipspringf      = 'BSColDFSS'
columnsection       = 'ColDSection'
columnrigidsection = 'ColRigidSection'
nIp_stif = 3;
nIp_flex = 5;
flexparams = {10000,1e-3}
-- Define elements
barslipbottom   = { 1, 'InterfaceElement2D', 1, 2, { {barslipspringf, {1,0,0} } },
                        { {barslipsectionf}}, {0,1,0},{-1,0,0} }
plasticcolumn1 = { 2, 'StiffnessBased2DBeamColumn', 2, 3, {columnsection, nIp_stif} }
flexcolumn      = { 3, 'FlexibilityBased2DBeamColumn', 3, 4, {columnsection,nIp_flex}, flexparams }
plasticcolumn2 = { 4, 'StiffnessBased2DBeamColumn',4, 5, {columnsection, nIp_stif} }
table.insert(elements, barslipbottom)
table.insert(elements, plasticcolumn1)
table.insert(elements, flexcolumn)
table.insert(elements, plasticcolumn2)
-- *******************************************************
-- Set section properties.
--
allsections = {
-- ------------
'ColDSection' , 'Fiber',
-- Tag, Area, y-loc, z-loc
{    'ColDCover' ,    5.1 ,    2.575    ,    0    ,
     'ColDCover' ,    5.1 ,   -2.575    ,    0    ,
     'ColDCover' ,    0.91375 ,    1.88125 ,    0    ,
     'ColDCover' ,    0.91375 ,    1.34375 ,    0    ,
     'ColDCover' ,    0.91375 ,    0.80625 ,    0    ,
     'ColDCover' ,    0.91375 ,    0.26875 ,    0    ,
     'ColDCover' ,    0.91375 ,   -0.26875    ,    0    ,
     'ColDCover' ,    0.91375 ,   -0.80625    ,    0    ,
     'ColDCover' ,    0.91375 ,   -1.34375    ,    0    ,
     'ColDCover' ,    0.91375 ,   -1.88125    ,    0    ,
     'ColDCore'  ,    2.31125 ,    1.88125 ,    0    ,
     'ColDCore'  ,    2.31125 ,    1.34375 ,    0    ,
     'ColDCore'  ,    2.31125 ,    0.80625 ,    0    ,
     'ColDCore'  ,    2.31125 ,    0.26875 ,    0    ,
     'ColDCore'  ,    2.31125 ,   -0.26875    ,    0    ,
     'ColDCore'  ,    2.31125 ,   -0.80625    ,    0    ,
     'ColDCore'  ,    2.31125 ,   -1.34375    ,    0    ,
     'ColDCore'  ,    2.31125 ,   -1.88125    ,    0    ,
     'ColDSteel' ,    0.147   ,    2.15     ,    0    ,
     'ColDSteel' ,    0.098   ,    0    ,    0    ,
     'ColDSteel' ,    0.147   ,   -2.15     ,    0    };
-- ------------
'BSColDFSection'   , 'Fiber',
{    'BSColDFCover'  ,    5.1 ,    2.575    ,    0    ,
     'BSColDFCover'  ,    5.1 ,   -2.575    ,    0    ,
     'BSColDFCover'  ,    0.91375 ,    1.88125 ,    0    ,
     'BSColDFCover'  ,    0.91375 ,    1.34375 ,    0    ,
     'BSColDFCover'  ,    0.91375 ,    0.80625 ,    0    ,
     'BSColDFCover'  ,    0.91375 ,    0.26875 ,    0    ,
     'BSColDFCover'  ,    0.91375 ,   -0.26875    ,    0    ,
     'BSColDFCover'  ,    0.91375 ,   -0.80625    ,    0    ,
     'BSColDFCover'  ,    0.91375 ,   -1.34375    ,    0    ,
     'BSColDFCover'  ,    0.91375 ,   -1.88125    ,    0    ,
     'BSColDFCore'   ,    2.31125 ,    1.88125 ,    0    ,
     'BSColDFCore'   ,    2.31125 ,    1.34375 ,    0    ,
     'BSColDFCore'   ,    2.31125 ,    0.80625 ,    0    ,
     'BSColDFCore' , 2.31125 ,    0.26875 ,    0    ,
     'BSColDFCore'   ,    2.31125 ,   -0.26875    ,    0    ,
     'BSColDFCore'   ,    2.31125 ,   -0.80625    ,    0    ,
     'BSColDFCore'   ,    2.31125 ,   -1.34375    ,    0    ,
     'BSColDFCore'   ,    2.31125 ,   -1.88125    ,    0    ,
     'BSColDFSteel'  ,    0.147   ,    2.15     ,    0    ,
     'BSColDFSteel'  ,    0.098   ,    0    ,    0    ,
     'BSColDFSteel'  ,    0.147   ,   -2.15     ,    0    }; }
-- *******************************************************
-- Set material properties.
concretemat = 'ConcreteLinearTensionSoftening'
steelmat    = 'GiuffreMenegottoPinto'
sspringmat  = 'Bilinear'
materials = {
{'ColDCover', 'anisotropicdamage2',2829,0,0.2,5.855E-08,1870,0.003248, 0.05168,38890000000,22.27,0.9999999};
{'ColDCore','anisotropicdamage2',1804,0,0.2,0.000006483,503.6,3.125E-07,0.4054,3159000000,102,0.9999999};
{'BSColDFCover','anisotropicdamage2',569.3, 0,0.2,1.689E-19, 371.8,0.832,1.461,9.529E+13,100.4,0.9999999};
{'BSColDFCore','anisotropicdamage2',396.7,0,0.2,0.0008326,124.8, 2.084E-07,1.61,8.856E+11,62.48,0.9999999   };
{'ColDSteel',steelmat,26500,0,87.5,0.01,15,0.925,0.15, 0, 55, 0, 55, 0},
{'BSColDFSteel' , steelmat, 6949 , 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSColDFSS' , sspringmat, 1690,   0, 78.2, 0.173,  0, 55,  0, 55},
}
-- ***********************************************************************
function dumptable(x)
    local result = '{'
    for i,v in ipairs(x) do
        if (type(v) == 'table') then
            result = result .. dumptable(v)
        else
            result = result .. v .. ','
        end
    end
    return result .. '}\n'
end
-- ------------
--print(dumptable(nodes))
--print(dumptable(elements))
--print(dumptable(materials))
--print(dumptable(allsections))
-- ***********************************************************************
-- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
model = StructureModel(2,3)
```

```
-- Assign all input data to Mercury
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(allsections)
model:addElements(elements)
-- ************************************************************************
-- constrain bottom nodes
model:constrainNode(1,1,1,1)
-- ************************************************************************
print("Transient analysis started\n")
earthquakeloading = LoadDescription()
earthquakeloading:addLoad({'groundmotion','NR_g_dt_0_01_OpneSees.txt,dt=0.01', 1, 386.4})
-- *****************************************************
displ = {}
function displpertime(time)
    dx5,dy5,dz5 = model:nodeDisplacements(5)
    table.insert(displ, dx5)
    print("Work\n");
end
--
react = {}
function reactpertime(time)
    fx1,fy1 = model:nodeRestoringForces(1)
    table.insert(react, fx1)
end
-- *******************************************************
solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=100000})
transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.2, 0.36, 0.7)
transientanalysis:addcallback(displpertime, "timestep")
transientanalysis:addcallback(reactpertime, "timestep")
model:setRayleighCoefficients(0.6318799279194399, 0.00015503501814608252)
transientanalysis:solve(7641)
-- *****************************************************
function writedata1(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 0) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
writedata1(displ,'Ex28HHTNodalDisp_5.dat')
writedata1(react,'Ex28HHTReact_1.dat')
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 ans constitutive model parameters used[1].

Nodal masses are used in the dynamic analysis, Table 10 to 12, where each beam has an added lead bundle masses at 4^{th} and 5^{th} 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

```
xxx
```

[1] Col: 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: modifed Kent-Park model, ModGMP: modified Giuffre-Monegotto-Pinto

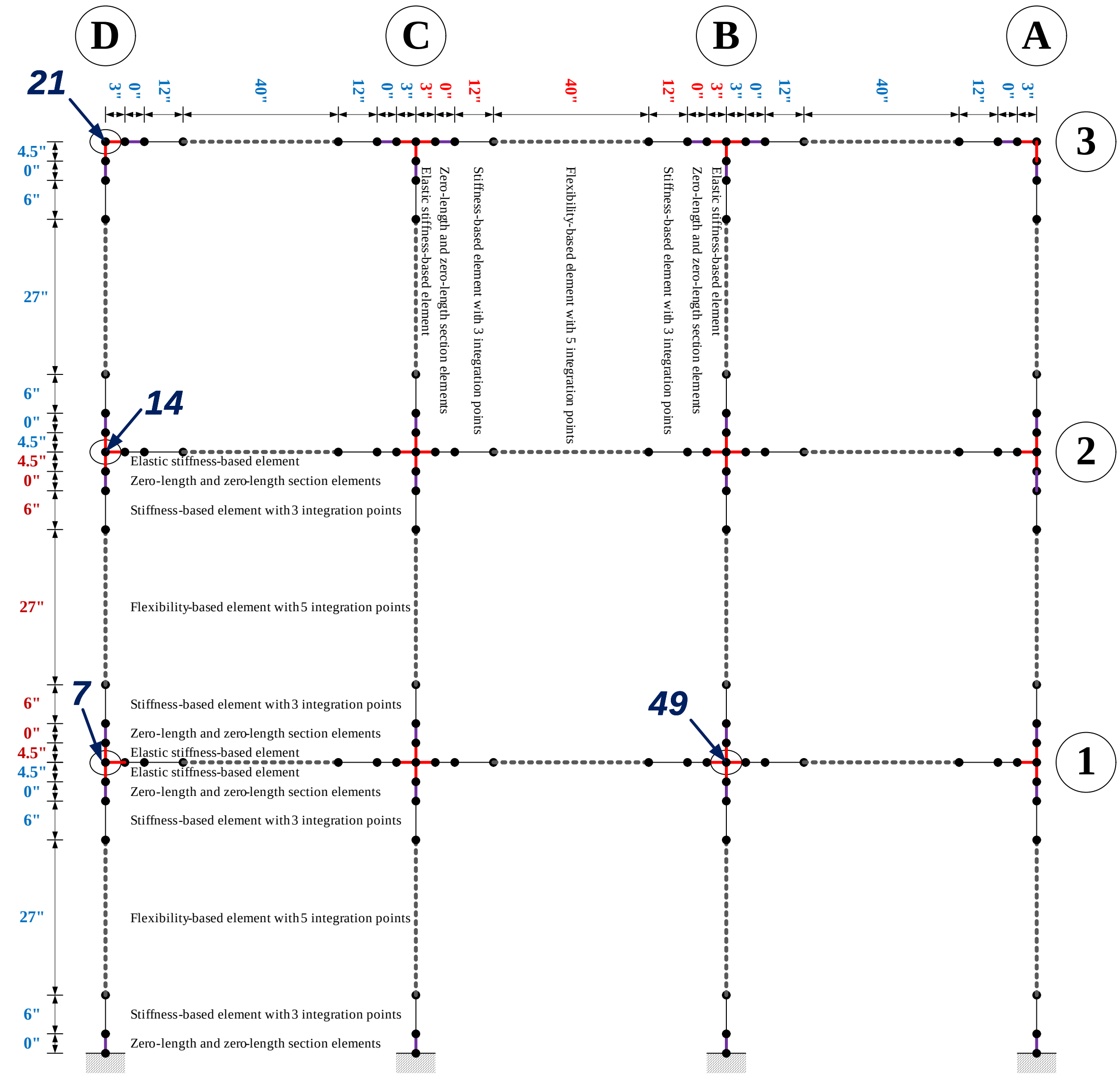


Figure 12: Numerical model for real-time hybrid simulation

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

Fiber name	Material	E_ts	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

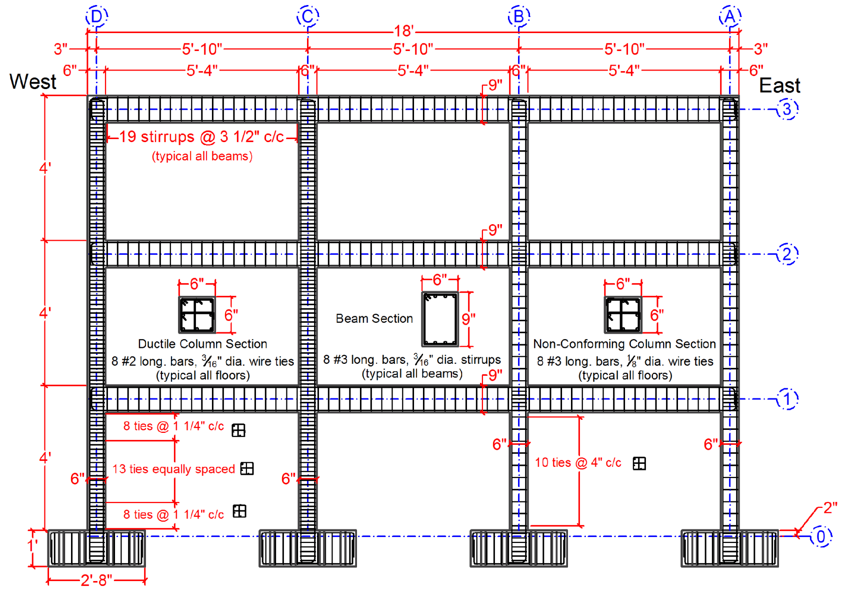


Figure 13: Reinforced concrete details, **?**

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

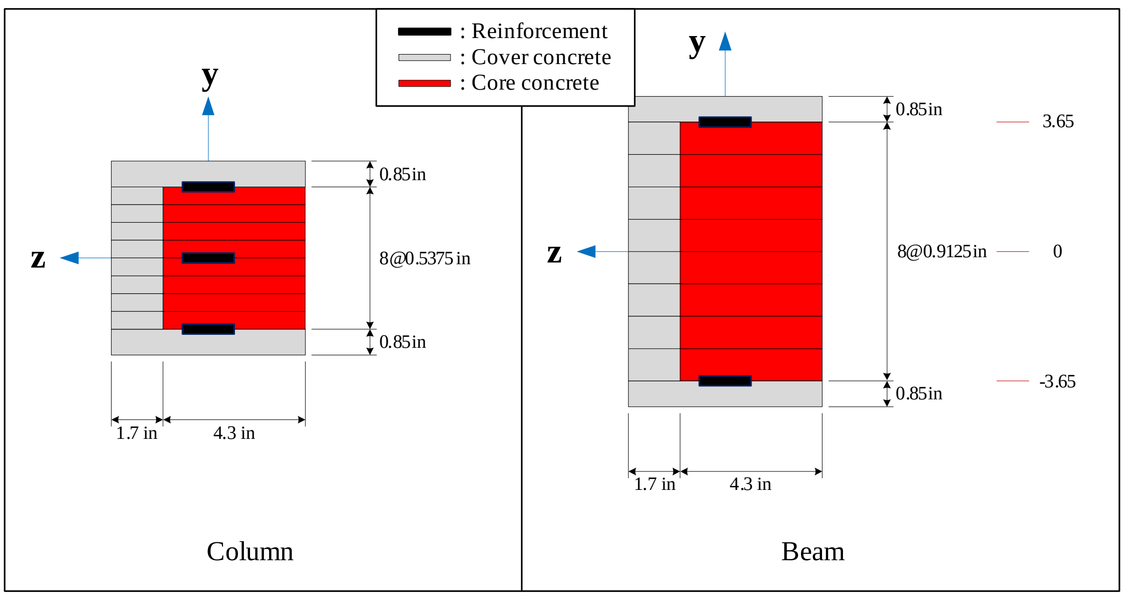


Figure 15: Section properties of Ex29

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	Elasitc	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++

```
 ***********************************************
earthquakefiles = {
     'Test16AccelHHalfYield.txt',
     'Test19AccelH1stCollapse.txt',
     'Test30AccelH2ndCollapse.txt',
     'Test32AccelH3rdCollapse.txt'
}
earthquakefile = earthquakefiles[2]
print('Excitation:',earthquakefile)
-- ************************************************************************
alpha  = -0.2
beta   = (1-alpha)*(1-alpha)/4
gamma  = (1-2*alpha)/2
-- alpha : alpha coefficient in HHT integration
-- beta  : beta coefficient in HHT integration
-- gamma : gamma coefficient in HHT integration
-- ************************************************************************
-- Multiple bay/floor building model
-- ************************************************************************
-- Column : first floor
--    o    (43.5)
--    |         Elastic stiffness-based beam-column
--    o    (39)
--              Zero-Length and zero-Length section elements(top bar slip and shear deformation)
--    o    (39)
--    |         Stiffness-based beam-column with 2 integration points
--    o    (33)
--    |
--    |         Flexibility-based beam-column with 5 integration points
--    |
--    o    (6)
--    |         Stiffness-based beam-column with 2 integration points
--    o    (0)
--              Zero-Length and zero-Length section elements(bottom bar slip and shear deformation)
--   _o_   (0)  (Fixed support)

-- ************************************************************************
-- Column : other floor
--    o    (48)
--    |         Elastic stiffness-based beam-column
--    o    (43.5)
--              Zero-Length and zero-Length section elements(top bar slip and shear deformation)
--    o    (43.5)
--    |         Stiffness-based beam-column with 2 integration points
--    o    (37.5)
--    |
--    |         Flexibility-based beam-column with 5 integration points
--    |
--    o    (10.5)
--    |         Stiffness-based beam-column with 2 integration points
--    o    (4.5)
--              Zero-Length and zero-Length section elements(bottom bar slip and shear deformation)
--    o    (4.5)
--    |         Elastic stiffness-based beam-column
--    o    (0)
-- ************************************************************************
-- Beam
--  o-----o     o-----o--------o-----o     o-----o
-- (0) a (3) b (3) c (15)  d  (55)e(67) f (67) g (70)
-- a: Elastic stiffness-based beam-column
-- b: Zero-Length and zero-Length section elements(left bar slip and shear deformation)
-- c: Stiffness-based beam-column with 2 integration points
-- d: Flexibility-based beam-column with 5 integration points
-- e: Stiffness-based beam-column with 2 integration points
-- f: Zero-Length and zero-Length section elements(right bar slip and shear deformation)
-- g: Elastic stiffness-based beam-column
-- ************************************************************************
-- Basic variables for node information
-- g      : gravity acceleration
-- nbays : Number of bays (nbays must be odd number)
-- nflrs : Number of floors
-- n1cnod: Number of nodes of the first column
-- nocnod: Number of nodes of other columns
-- nbmnod: Number of nodes of beams
-- c1len : Total length of the first column
-- colen : Total length of other columns
-- bmlen : Total length of beams
-- c1nod : Coordinate information on the first column
-- conod : Coordinate information on ohter columns
-- bmnod : Coordinate information on beams
-- cd1nodmass : Nodal mass of the first ductile column
-- cdonodmass : Nodal mass of other ductile columns
-- cnd1nodmass: Nodal mass of the first shear critical column
-- cndonodmass: Nodal mass of other shear critical column
-- nIp_elas    : Number of integration points of elastic stiffness-based beam-column
-- nIp_stif    : Number of integration points of nonlinear stiffness-based beam-column
-- nIp_flex    : Number of integration points of nonlinear flexibility-based beam-column3
-- flexparams : Iteration config for flexibility elements
-- flexparams = {number of iterations, tolerance}
-- $$$ start of node and element information $$$
g      = 386.4
nbays = 3;  -- nbays must be odd number.
nflrs = 3;
n1cnod= 7;
nocnod= 8;
nbmnod= 8;
c1len = 43.5;
colen = 48;
bmlen = 70;
c1nod = {0,0,6,33,39,39,43.5};
conod = {0,4.5,4.5,10.5,37.5,43.5,43.5,48};
bmnod = {0,3,3,15,55,67,67,70};
bmaddWeight = 3
bmaddMass    = bmaddWeight/g
```

```
-- ------------
c1nodmass = {0.0, 2.50712E-05, 0.000137891, 0.000137891,  2.50712E-05,  1.88034E-05,  7.52135E-05}
conodmass = {0.0, 1.88034E-05, 2.50712E-05, 0.000137891,  0.000137891,  2.50712E-05,  1.88034E-05,  6.26779E-05}
bmnodmass = {0.0, 1.88034E-05, 7.52135E-05, 0.000325925+bmaddMass, 0.000325925+bmaddMass,
             7.52135E-05, 1.88034E-05, 0.0}
-- ------------
nIp_elas = 1;
nIp_stif = 3;
nIp_flex = 5;
-- ------------
flexparams = {1000,1e-6}
-- $$$ END of node and element information $$$
-- *********************************************************************
-- Set nodes and elements.
--
--
nodes    = {}
elements = {}
-- ------------
function nodename(bay,floor)
    return string.format('node_%d_%d', bay, floor)
end
-- ------------
function columnnodename(bay,floor,subsection)
    if (subsection == 1 and floor > 1) then
--      return columnnodename(bay,floor-1,nocnod)
        if (floor == 2) then
            return columnnodename(bay,floor-1,n1cnod)
        else
            return columnnodename(bay,floor-1,nocnod)
        end
    else
        return string.format('colnode_%d_%d_%d',bay,floor,subsection)
    end
end
-- ------------
function beamnodename(bay,floor,subsection)
    -- section 1 nodes re-use the column nodes
    if (subsection == 1) then
        return columnnodename(bay,floor,1)
    else
        return string.format('beamnode_%d_%d_%d',bay,floor,subsection)
    end
end
-- ------------

-- create ductile column node coordinates
ncold = (nbays+1)/2;
for colbay = 1,ncold do
    for colfloor = 1,nflrs do
        x = (colbay-1)*bmlen
        nodeyfirst  = c1nod
        nodeyhigher = conod
        -- create nodes
        if (colfloor==1) then
            bottomnode = columnnodename(colbay,colfloor,1)
            for k=1,n1cnod do
                y       = nodeyfirst[k]
                masscd1 = c1nodmass[k]
                curnode = { columnnodename(colbay,colfloor,k), x,y, 'mass', masscd1, masscd1, 0.1*masscd1  }
                table.insert(nodes,curnode)
            end
        else
            bottomnode = columnnodename(colbay,colfloor-1,nocnod)
            for k=2,nocnod do
                y       = c1len + colen * (colfloor-2) + nodeyhigher[k]
                masscdo = conodmass[k]
                curnode = { columnnodename(colbay,colfloor,k), x,y, 'mass', masscdo, masscdo, 0.1*masscdo }
                table.insert(nodes,curnode)
            end
        end
        -- figure out section names
        barslipsectionf    = 'BSColDFSection'
        barslipspringf     = 'BSColDFSS'
        columnsection      = 'ColDSection'
    barslipspring      = 'BSColDSS'
    barslipsection     = 'BSColDSection'
    columnrigidsection = 'ColRigidSection'
    if (colfloor==1) then
        node1 = columnnodename(colbay,colfloor,1)
            node2 = columnnodename(colbay,colfloor,2)
            node3 = columnnodename(colbay,colfloor,3)
            node4 = columnnodename(colbay,colfloor,4)
            node5 = columnnodename(colbay,colfloor,5)
            node6 = columnnodename(colbay,colfloor,6)
            node7 = columnnodename(colbay,colfloor,7)
            -- Define elements
            barslipbottom  = { string.format('columnbsb_%d_%d',colbay,colfloor), 'InterfaceElement2D', node1, node2,
                      { {barslipspringf, {1,0,0} } },{ {barslipsectionf}}, {0,1,0},{-1,0,0} }
          plasticcolumn1 = { string.format('columnpl1_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                             node2, node3, {columnsection, nIp_stif} }
          flexcolumn     = { string.format('columnflx_%d_%d',colbay,colfloor), 'FlexibilityBased2DBeamColumn',
                             node3, node4, {columnsection,nIp_flex}, flexparams }
          plasticcolumn2 = { string.format('columnpl2_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                             node4, node5, {columnsection, nIp_stif} }
          barsliptop     = { string.format('columnbst_%d_%d',colbay,colfloor), 'InterfaceElement2D', node5, node6,
                             { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
            rigidcolumntop = { string.format('columnrdt_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                               node6, node7, {columnrigidsection, nIp_elas} }
            table.insert(elements, barslipbottom)
            table.insert(elements, plasticcolumn1)
            table.insert(elements, flexcolumn)
            table.insert(elements, plasticcolumn2)
            table.insert(elements, barsliptop)
            table.insert(elements, rigidcolumntop)
    else
        if (colfloor == 2) then
```

```
                node1 = columnnodename(colbay, colfloor-1, n1cnod)
            else
                node1 = columnnodename(colbay, colfloor-1, nocnod)
            end
            node2 = columnnodename(colbay,colfloor,2)
            node3 = columnnodename(colbay,colfloor,3)
            node4 = columnnodename(colbay,colfloor,4)
            node5 = columnnodename(colbay,colfloor,5)
            node6 = columnnodename(colbay,colfloor,6)
            node7 = columnnodename(colbay,colfloor,7)
            node8 = columnnodename(colbay,colfloor,8)
            -- Define elements
            rigidcolumnbottom = { string.format('columnrdb_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                                  node1, node2, {columnrigidsection, nIp_elas} }
            barslipbottom     = { string.format('columnbsb_%d_%d',colbay,colfloor), 'InterfaceElement2D', node2,
                                 node3, { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
          plasticcolumn1    = { string.format('columnpl1_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                               node3, node4, {columnsection, nIp_stif} }
          flexcolumn        = { string.format('columnflx_%d_%d',colbay,colfloor), 'FlexibilityBased2DBeamColumn',
                               node4, node5, {columnsection,nIp_flex}, flexparams }
          plasticcolumn2    = { string.format('columnpl2_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                               node5, node6, {columnsection, nIp_stif} }
          barsliptop        = { string.format('columnbst_%d_%d',colbay,colfloor), 'InterfaceElement2D', node6,
                               node7,{ {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
            rigidcolumntop    = { string.format('columnrdt_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                                  node7, node8, {columnrigidsection, nIp_elas} }
            table.insert(elements, rigidcolumnbottom)
            table.insert(elements, barslipbottom)
            table.insert(elements, plasticcolumn1)
            table.insert(elements, flexcolumn)
            table.insert(elements, plasticcolumn2)
            table.insert(elements, barsliptop)
            table.insert(elements, rigidcolumntop)
    end
    end
end

-- create shear critical column node coordinates
for colbay = ncold+1,nbays+1 do
    for colfloor = 1,nflrs do
        x = (colbay-1)*bmlen
        nodeyfirst  = c1nod
        nodeyhigher = conod
        -- create nodes
        if (colfloor==1) then
            bottomnode = columnnodename(colbay,colfloor,1)
            for k=1,n1cnod do
                y        = nodeyfirst[k]
                masscnd1 = c1nodmass[k]
                curnode  = { columnnodename(colbay,colfloor,k), x,y, 'mass', masscnd1, masscnd1, 0.1*masscnd1 }
                table.insert(nodes,curnode)
            end
        else
            bottomnode = columnnodename(colbay,colfloor-1,nocnod)
            for k=2,nocnod do
                y        = c1len + colen * (colfloor-2) + nodeyhigher[k]
                masscndo = conodmass[k]
                curnode  = { columnnodename(colbay,colfloor,k), x,y, 'mass', masscndo, masscndo, 0.1*masscndo }
                table.insert(nodes,curnode)
            end
        end
                -- figure out section names
        barslipsectionf    = 'BSColNDFSection'
        barslipspringf     = 'BSColNDFSS'
        columnsection      = 'ColNDSection'
    barslipspring       = 'BSColNDSS'
    barslipsection      = 'BSColNDSection'
    columnrigidsection = 'ColRigidSection'
    if (colfloor==1) then
        node1 = columnnodename(colbay,colfloor,1)
            node2 = columnnodename(colbay,colfloor,2)
            node3 = columnnodename(colbay,colfloor,3)
            node4 = columnnodename(colbay,colfloor,4)
            node5 = columnnodename(colbay,colfloor,5)
            node6 = columnnodename(colbay,colfloor,6)
            node7 = columnnodename(colbay,colfloor,7)
            -- Define elements
            barslipbottom  = { string.format('columnbsb_%d_%d',colbay,colfloor), 'InterfaceElement2D',
                               node1, node2, { {barslipspringf, {1,0,0} } }, { {barslipsectionf}}, {0,1,0},{-1,0,0} }
          plasticcolumn1 = { string.format('columnpl1_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                            node2, node3, {columnsection, nIp_stif} }
          flexcolumn     = { string.format('columnflx_%d_%d',colbay,colfloor), 'FlexibilityBased2DBeamColumn',
                            node3, node4, {columnsection,nIp_flex}, flexparams }
          plasticcolumn2 = { string.format('columnpl2_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                            node4, node5, {columnsection, nIp_stif} }
          barsliptop     = { string.format('columnbst_%d_%d',colbay,colfloor), 'InterfaceElement2D', node5, node6,
                            { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
            rigidcolumntop = { string.format('columnrdt_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                               node6, node7, {columnrigidsection, nIp_elas} }
            table.insert(elements, barslipbottom)
            table.insert(elements, plasticcolumn1)
            table.insert(elements, flexcolumn)
            table.insert(elements, plasticcolumn2)
            table.insert(elements, barsliptop)
            table.insert(elements, rigidcolumntop)
    else
        if (colfloor == 2) then
                node1 = columnnodename(colbay, colfloor-1, n1cnod)
            else
                node1 = columnnodename(colbay, colfloor-1, nocnod)
            end
            node2 = columnnodename(colbay,colfloor,2)
            node3 = columnnodename(colbay,colfloor,3)
            node4 = columnnodename(colbay,colfloor,4)
            node5 = columnnodename(colbay,colfloor,5)
            node6 = columnnodename(colbay,colfloor,6)
            node7 = columnnodename(colbay,colfloor,7)
```

```
            node8 = columnnodename(colbay,colfloor,8)
            -- Define elements
            rigidcolumnbottom = { string.format('columnrdb_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                                  node1, node2, {columnrigidsection, nIp_elas} }
            barslipbottom     = { string.format('columnbsb_%d_%d',colbay,colfloor), 'InterfaceElement2D',
                                 node2, node3, { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
          plasticcolumn1    = { string.format('columnpl1_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                              node3, node4, {columnsection, nIp_stif} }
          flexcolumn        = { string.format('columnflx_%d_%d',colbay,colfloor), 'FlexibilityBased2DBeamColumn',
                              node4, node5, {columnsection,nIp_flex}, flexparams }
          plasticcolumn2    = { string.format('columnpl2_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                              node5, node6, {columnsection, nIp_stif} }
          barsliptop        = { string.format('columnbst_%d_%d',colbay,colfloor), 'InterfaceElement2D', node6, node7,
                              { {barslipspring, {1,0,0} } }, { {barslipsection}}, {0,1,0},{-1,0,0} }
            rigidcolumntop    = { string.format('columnrdt_%d_%d',colbay,colfloor), 'StiffnessBased2DBeamColumn',
                                 node7, node8, {columnrigidsection, nIp_elas} }
            table.insert(elements, rigidcolumnbottom)
            table.insert(elements, barslipbottom)
            table.insert(elements, plasticcolumn1)
            table.insert(elements, flexcolumn)
            table.insert(elements, plasticcolumn2)
            table.insert(elements, barsliptop)
            table.insert(elements, rigidcolumntop)
    end
    end
end

-- create beam node coordinates
for beambay = 1,nbays do
    for beamfloor = 2,nflrs+1 do
    y=c1len+colen*(beamfloor-2)
        -- create nodes
        for k=2,nbmnod-1 do
            x       = (beambay-1)*bmlen + bmnod[k]
      massbm= bmnodmass[k]
            curnode = { beamnodename(beambay,beamfloor,k), x ,y, 'mass', massbm, massbm, 0.1*massbm }
            table.insert(nodes,curnode)
        end
        -- figure out section names
        beamsection      = 'BeamSection'
    barslipsection    = 'BSBeamSection'
        beamrigidsection = 'BeamRigidSection'
        barslipspring    = 'BSBeamSS'
        -- figure out noda tag for node connectivity
        node1 = beamnodename(beambay,beamfloor,1)
        node2 = beamnodename(beambay,beamfloor,2)
        node3 = beamnodename(beambay,beamfloor,3)
        node4 = beamnodename(beambay,beamfloor,4)
        node5 = beamnodename(beambay,beamfloor,5)
    node6 = beamnodename(beambay,beamfloor,6)
    node7 = beamnodename(beambay,beamfloor,7)
    node8 = beamnodename(beambay+1, beamfloor,1)
    -- Define elements
    rigidbeamleft = { string.format('beamrdl_%d_%d',beambay,beamfloor),
                    'StiffnessBased2DBeamColumn', node1, node2, {beamrigidsection, nIp_elas} }
        barslipleft   = { string.format('beambsl_%d_%d',beambay,beamfloor),
                          'InterfaceElement2D', node2, node3,
                            { {barslipspring, {0,1,0} } }, { {barslipsection}}, {1,0,0},{0,1,0} }
        plasticbeam1  = { string.format('beampl1_%d_%d',beambay,beamfloor),
                            'StiffnessBased2DBeamColumn', node3, node4, {beamsection, nIp_stif} }
        flexbeam      = { string.format('beamflx_%d_%d',beambay,beamfloor),
                            'FlexibilityBased2DBeamColumn', node4, node5, {beamsection,nIp_flex}, flexparams }
        plasticbeam2  = { string.format('beampl2_%d_%d',beambay,beamfloor),
                            'StiffnessBased2DBeamColumn',node5, node6, {beamsection, nIp_stif} }
        barslipright  = { string.format('beambsr_%d_%d',beambay,beamfloor),
                            'InterfaceElement2D', node6, node7, { {barslipspring, {0,1,0} } },
                            { {barslipsection}}, {1,0,0},{0,1,0} }
        rigidbeamright= { string.format('beamrdr_%d_%d',beambay,beamfloor), 'StiffnessBased2DBeamColumn',
                            node7, node8, {beamrigidsection, nIp_elas} }
        table.insert(elements, rigidbeamleft)
        table.insert(elements, barslipleft)
        table.insert(elements, plasticbeam1)
        table.insert(elements, flexbeam)
        table.insert(elements, plasticbeam2)
        table.insert(elements, barslipright)
        table.insert(elements, rigidbeamright)
    end
end
 -- ***********************************************************************
 -- Set section properties.
 --
 --
allsections = {
-- ------------
'ColDSection' , 'Fiber',
-- Tag, Area, y-loc, z-loc
{   'ColDCover' ,  5.1 ,   2.575   ,  0   ,
    'ColDCover' ,  5.1 ,  -2.575   ,  0   ,
    'ColDCover' ,  0.91375 ,  1.88125 ,  0   ,
    'ColDCover' ,  0.91375 ,  1.34375 ,  0   ,
    'ColDCover' ,  0.91375 ,  0.80625 ,  0   ,
    'ColDCover' ,  0.91375 ,  0.26875 ,  0   ,
    'ColDCover' ,  0.91375 ,  -0.26875   ,  0   ,
    'ColDCover' ,  0.91375 ,  -0.80625   ,  0   ,
    'ColDCover' ,  0.91375 ,  -1.34375   ,  0   ,
    'ColDCover' ,  0.91375 ,  -1.88125   ,  0   ,
    'ColDCore'  ,  2.31125 ,  1.88125 ,  0   ,
    'ColDCore'  ,  2.31125 ,  1.34375 ,  0   ,
    'ColDCore'  ,  2.31125 ,  0.80625 ,  0   ,
    'ColDCore'  ,  2.31125 ,  0.26875 ,  0   ,
    'ColDCore'  ,  2.31125 ,  -0.26875   ,  0   ,
    'ColDCore'  ,  2.31125 ,  -0.80625   ,  0   ,
    'ColDCore'  ,  2.31125 ,  -1.34375   ,  0   ,
    'ColDCore'  ,  2.31125 ,  -1.88125   ,  0   ,
    'ColDSteel' ,  0.147   ,  2.15    ,  0   ,
    'ColDSteel' ,  0.098   ,  0   ,  0   ,
    'ColDSteel' ,  0.147   ,  -2.15   ,  0   };
```

```
-- ------------
'BSColDSection'    , 'Fiber',
{    'BSColDCover'    ,    5.1 ,    2.575    ,    0    ,
     'BSColDCover'    ,    5.1 ,   -2.575    ,    0    ,
     'BSColDCover'    ,    0.91375 ,    1.88125 ,    0    ,
     'BSColDCover'    ,    0.91375 ,    1.34375 ,    0    ,
     'BSColDCover'    ,    0.91375 ,    0.80625 ,    0    ,
     'BSColDCover'    ,    0.91375 ,    0.26875 ,    0    ,
     'BSColDCover'    ,    0.91375 ,   -0.26875    ,    0    ,
     'BSColDCover'    ,    0.91375 ,   -0.80625    ,    0    ,
     'BSColDCover'    ,    0.91375 ,   -1.34375    ,    0    ,
     'BSColDCover'    ,    0.91375 ,   -1.88125    ,    0    ,
     'BSColDCore'     ,    2.31125 ,    1.88125 ,    0    ,
     'BSColDCore'     ,    2.31125 ,    1.34375 ,    0    ,
     'BSColDCore'     ,    2.31125 ,    0.80625 ,    0    ,
     'BSColDCore'     ,    2.31125 ,    0.26875 ,    0    ,
     'BSColDCore'     ,    2.31125 ,   -0.26875    ,    0    ,
     'BSColDCore'     ,    2.31125 ,   -0.80625    ,    0    ,
     'BSColDCore'     ,    2.31125 ,   -1.34375    ,    0    ,
     'BSColDCore'     ,    2.31125 ,   -1.88125    ,    0    ,
     'BSColDSteel'    ,    0.147   ,    2.15    ,    0    ,
     'BSColDSteel'    ,    0.098   ,    0    ,    0    ,
     'BSColDSteel'    ,    0.147   ,   -2.15    ,    0    };
-- ------------
'BSColDFSection'   , 'Fiber',
{    'BSColDFCover'   ,    5.1 ,    2.575    ,    0    ,
     'BSColDFCover'   ,    5.1 ,   -2.575    ,    0    ,
     'BSColDFCover'   ,    0.91375 ,    1.88125 ,    0    ,
     'BSColDFCover'   ,    0.91375 ,    1.34375 ,    0    ,
     'BSColDFCover'   ,    0.91375 ,    0.80625 ,    0    ,
     'BSColDFCover'   ,    0.91375 ,    0.26875 ,    0    ,
     'BSColDFCover'   ,    0.91375 ,   -0.26875    ,    0    ,
     'BSColDFCover'   ,    0.91375 ,   -0.80625    ,    0    ,
     'BSColDFCover'   ,    0.91375 ,   -1.34375    ,    0    ,
     'BSColDFCover'   ,    0.91375 ,   -1.88125    ,    0    ,
     'BSColDFCore'      , 2.31125 ,    1.88125 ,    0    ,
     'BSColDFCore'      , 2.31125 ,    1.34375 ,    0    ,
     'BSColDFCore'      , 2.31125 ,    0.80625 ,    0    ,
     'BSColDFCore'    ,    2.31125 ,    0.26875 ,    0    ,
     'BSColDFCore'      , 2.31125 ,   -0.26875    ,    0    ,
     'BSColDFCore'      , 2.31125 ,   -0.80625    ,    0    ,
     'BSColDFCore'      , 2.31125 ,   -1.34375    ,    0    ,
     'BSColDFCore'      , 2.31125 ,   -1.88125    ,    0    ,
     'BSColDFSteel'   ,    0.147   ,    2.15    ,    0    ,
     'BSColDFSteel'   ,    0.098   ,    0    ,    0    ,
     'BSColDFSteel'   ,    0.147   ,   -2.15    ,    0    };
-- ------------
'ColNDSection'     , 'Fiber',
{    'ColNDCover'     ,    5.1 ,    2.575    ,    0    ,
     'ColNDCover'     ,    5.1 ,   -2.575    ,    0    ,
     'ColNDCover'     ,    0.91375 ,    1.88125 ,    0    ,
     'ColNDCover'     ,    0.91375 ,    1.34375 ,    0    ,
     'ColNDCover'     ,    0.91375 ,    0.80625 ,    0    ,
     'ColNDCover'     ,    0.91375 ,    0.26875 ,    0    ,
     'ColNDCover'     ,    0.91375 ,   -0.26875    ,    0    ,
     'ColNDCover'     ,    0.91375 ,   -0.80625    ,    0    ,
     'ColNDCover'     ,    0.91375 ,   -1.34375    ,    0    ,
     'ColNDCover'     ,    0.91375 ,   -1.88125    ,    0    ,
     'ColNDCore'    , 2.31125 ,    1.88125 ,    0    ,
     'ColNDCore'    , 2.31125 ,    1.34375 ,    0    ,
     'ColNDCore'    , 2.31125 ,    0.80625 ,    0    ,
     'ColNDCore'    , 2.31125 ,    0.26875 ,    0    ,
     'ColNDCore'    , 2.31125 ,   -0.26875    ,    0    ,
     'ColNDCore'    , 2.31125 ,   -0.80625    ,    0    ,
     'ColNDCore'    , 2.31125 ,   -1.34375    ,    0    ,
     'ColNDCore'    , 2.31125 ,   -1.88125    ,    0    ,
     'ColNDSteel'     ,    0.33    ,    2.15    ,    0    ,
     'ColNDSteel'     ,    0.22    ,    0    ,    0    ,
     'ColNDSteel'     ,    0.33    ,   -2.15    ,    0    };
-- ------------
'BSColNDSection'   , 'Fiber',
{    'BSColNDCover'   ,    5.1 ,    2.575    ,    0    ,
     'BSColNDCover'   ,    5.1 ,   -2.575    ,    0    ,
     'BSColNDCover'   ,    0.91375 ,    1.88125 ,    0    ,
     'BSColNDCover'   ,    0.91375 ,    1.34375 ,    0    ,
     'BSColNDCover'   ,    0.91375 ,    0.80625 ,    0    ,
     'BSColNDCover'   ,    0.91375 ,    0.26875 ,    0    ,
     'BSColNDCover'   ,    0.91375 ,   -0.26875    ,    0    ,
     'BSColNDCover'   ,    0.91375 ,   -0.80625    ,    0    ,
     'BSColNDCover'   ,    0.91375 ,   -1.34375    ,    0    ,
     'BSColNDCover'   ,    0.91375 ,   -1.88125    ,    0    ,
     'BSColNDCore'      , 2.31125 ,    1.88125 ,    0    ,
     'BSColNDCore'      , 2.31125 ,    1.34375 ,    0    ,
     'BSColNDCore'      , 2.31125 ,    0.80625 ,    0    ,
     'BSColNDCore'      , 2.31125 ,    0.26875 ,    0    ,
     'BSColNDCore'      , 2.31125 ,   -0.26875    ,    0    ,
     'BSColNDCore'      , 2.31125 ,   -0.80625    ,    0    ,
     'BSColNDCore'      , 2.31125 ,   -1.34375    ,    0    ,
     'BSColNDCore'      , 2.31125 ,   -1.88125    ,    0    ,
     'BSColNDSteel'   ,    0.33    ,    2.15    ,    0    ,
     'BSColNDSteel'   ,    0.22    ,    0    ,    0    ,
     'BSColNDSteel'   ,    0.33    ,   -2.15    ,    0    };
-- ------------
'BSColNDFSection'  , 'Fiber',
{    'BSColNDFCover'  ,    5.1 ,    2.575    ,    0    ,
     'BSColNDFCover'  ,    5.1 ,   -2.575    ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,    1.88125 ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,    1.34375 ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,    0.80625 ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,    0.26875 ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,   -0.26875    ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,   -0.80625    ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,   -1.34375    ,    0    ,
     'BSColNDFCover'  ,    0.91375 ,   -1.88125    ,    0    ,
     'BSColNDFCore'   ,    2.31125 ,    1.88125 ,    0    ,
     'BSColNDFCore'   ,    2.31125 ,    1.34375 ,    0    ,
     'BSColNDFCore'   ,    2.31125 ,    0.80625 ,    0    ,
```

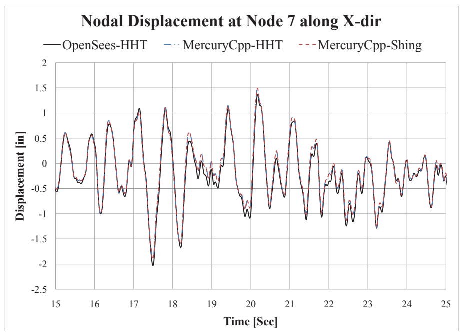
```
    'BSColNDFCore'  ,  2.31125 ,  0.26875 ,  0  ,
    'BSColNDFCore'  ,  2.31125 ,  -0.80625   ,  0  ,
    'BSColNDFCore'  ,  2.31125 ,  -0.26875   ,  0  ,
    'BSColNDFCore'  ,  2.31125 ,  -1.34375   ,  0  ,
    'BSColNDFCore'  ,  2.31125 ,  -1.88125   ,  0  ,
    'BSColNDFSteel' ,  0.33    ,  2.15   ,  0  ,
    'BSColNDFSteel' ,  0.22    ,  0   ,  0  ,
    'BSColNDFSteel' ,  0.33    ,  -2.15  ,  0  };
-- ------------
'BeamSection'     , 'Fiber',
{   'BeamCover' ,  5.1 ,  4.075  ,  0  ,
    'BeamCover' ,  5.1 ,  -4.075  ,  0  ,
    'BeamCover' ,  1.55125 ,  3.19375 ,  0  ,
    'BeamCover' ,  1.55125 ,  2.28125 ,  0  ,
    'BeamCover' ,  1.55125 ,  1.36875 ,  0  ,
    'BeamCover' ,  1.55125 ,  0.45625 ,  0  ,
    'BeamCover' ,  1.55125 ,  -0.45625   ,  0 ,
    'BeamCover' ,  1.55125 ,  -1.36875   ,  0 ,
    'BeamCover' ,  1.55125 ,  -2.28125   ,  0 ,
    'BeamCover' ,  1.55125 ,  -3.19375   ,  0 ,
    'BeamCore'  ,  3.92375 ,  3.19375 ,  0  ,
    'BeamCore'  ,  3.92375 ,  2.28125 ,  0  ,
    'BeamCore'  ,  3.92375 ,  1.36875 ,  0  ,
    'BeamCore'  ,  3.92375 ,  0.45625 ,  0  ,
    'BeamCore'  ,  3.92375 ,  -0.45625   ,  0  ,
    'BeamCore'  ,  3.92375 ,  -1.36875   ,  0  ,
    'BeamCore'  ,  3.92375 ,  -2.28125   ,  0  ,
    'BeamCore'  ,  3.92375 ,  -3.19375   ,  0  ,
    'BeamSteel' ,  0.44    ,  3.65   ,  0  ,
    'BeamSteel' ,  0.44    ,  -3.65  ,  0  };
-- ------------
'BSBeamSection'    , 'Fiber',
{   'BSBeamCover'  ,  5.1 ,  4.075  ,  0  ,
    'BSBeamCover'  ,  5.1 ,  -4.075  ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  3.19375 ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  2.28125 ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  1.36875 ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  0.45625 ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  -0.45625   ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  -1.36875   ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  -2.28125   ,  0  ,
    'BSBeamCover'  ,  1.55125 ,  -3.19375   ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  3.19375 ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  2.28125 ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  1.36875 ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  0.45625 ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  -0.45625   ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  -1.36875   ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  -2.28125   ,  0  ,
    'BSBeamCore'   ,  3.92375 ,  -3.19375   ,  0  ,
    'BSBeamSteel'  ,  0.44    ,  3.65   ,  0  ,
    'BSBeamSteel'  ,  0.44    ,  -3.65  ,  0  };
-- ------------
'ColRigidSection' , 'general',
{   'ColRigid'  ,  36  ,  108 };
-- ------------
'BeamRigidSection', 'general',
{   'BeamRigid' ,  54  ,  364.5  };
}
 -- ***********************************************************************
 -- Set material properties.
 --
 --
concretemat = 'ConcreteLinearTensionSoftening'
steelmat    = 'GiuffreMenegottoPinto'
sspringmat  = 'Bilinear'
materials = {
-- Concrete (ConcreteLinearTensionSoftening)
--          Tag,     mat_type,         Ets,  density,   sc,      ec,   scu,      ecu, lam,          st
{'ColDCover'    , concretemat, 549.2307692, 0, -3.57, -0.0026 , -1.19, -0.0078 , 0.3, 0.448121077},
{'ColDCore'     , concretemat, 549.4505495, 0, -7.5 , -0.00546, -7.35, -0.01638, 0.3, 0.649519053},
{'BSColDCover'  , concretemat, 105        , 0, -3.57, -0.0136 , -1.19, -0.0408 , 0.3, 0.448121077},
{'BSColDCore'   , concretemat, 104.1666667, 0, -7.5 , -0.0288 , -7.35, -0.0864 , 0.3, 0.649519053},
{'BSColDFCover' , concretemat, 105        , 0, -3.57, -0.0136 , -1.19, -0.0408 , 0.3, 0.448121077},
{'BSColDFCore'  , concretemat, 104.1666667, 0, -7.5 , -0.0288 , -6.75, -0.0864 , 0.3, 0.649519053},
{'ColNDCover'   , concretemat, 549.2307692, 0, -3.57, -0.0026 , -1.19, -0.0078 , 0.3, 0.448121077},
{'ColNDCore'    , concretemat, 549.2957746, 0, -3.9 , -0.00284, -3.51, -0.00852, 0.3, 0.46837485 },
{'BSColNDCover' , concretemat, 105        , 0, -3.57, -0.0136 , -1.19, -0.0408 , 0.3, 0.448121077},
{'BSColNDCore'  , concretemat, 106.8493151, 0, -3.9 , -0.0146 , -3.51, -0.0438 , 0.3, 0.46837485 },
{'BSColNDFCover', concretemat, 144.2424242, 0, -3.57, -0.0099 , -1.19, -0.0297 , 0.3, 0.448121077},
{'BSColNDFCore' , concretemat, 106.8493151, 0, -3.9 , -0.0146 , -3.51, -0.0438 , 0.3, 0.46837485 },
{'BeamCover'    , concretemat, 549.2307692, 0, -3.57, -0.0026 , -1.19, -0.0078 , 0.3, 0.448121077},
{'BeamCore'     , concretemat, 549.2957746, 0, -3.9 , -0.00284, -3.51, -0.00852, 0.3, 0.46837485 },
{'BSBeamCover'  , concretemat, 105        , 0, -3.57, -0.0136 , -1.19, -0.0408 , 0.3, 0.448121077},
{'BSBeamCore'   , concretemat, 106.8493151, 0, -3.9 , -0.0146 , -3.51, -0.0438 , 0.3, 0.46837485 },
-- Steel (GiuffreMenegottoPinto)
--         Tag, mat_type,      E,    density,   sy,    b, R0,   cR1,  cR2,a1, a2,a3, a4, s_init
{'ColDSteel'    , steelmat, 26500, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSColDSteel'  , steelmat, 5067 , 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSColDFSteel' , steelmat, 6949 , 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'ColNDSteel'   , steelmat, 27300, 0, 80  , 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSColNDSteel' , steelmat, 5220 , 0, 80  , 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSColNDFSteel', steelmat, 5220 , 0, 80  , 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BeamSteel'    , steelmat, 27300, 0, 80  , 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
{'BSBeamSteel'  , steelmat, 5220 , 0, 80  , 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
-- Shear spring (Bilinear)
--       Tag,    mat_type,     E, density,   sy,     b, a1, a2, a3, a4
{'BSColDSS'   , sspringmat, 1690,       0, 78.2, 0.173,  0, 55,  0, 55},
{'BSColDFSS'  , sspringmat, 1690,       0, 78.2, 0.173,  0, 55,  0, 55},
{'BSColNDSS'  , sspringmat, 1690,       0, 78.2, 0.173,  0, 55,  0, 55},
{'BSColNDFSS' , sspringmat, 1690,       0, 78.2, 0.173,  0, 55,  0, 55},
{'BSBeamSS'   , sspringmat, 1545,       0, 75.8, 0.396,  0, 55,  0, 55},
-- Rigid element (Elastic)
--        Tag, mat_type,          E, density
{'ColRigid' , 'Elastic', 10000000000, 0},
{'BeamRigid', 'Elastic', 10000000000, 0}
}
```

```
-- ***********************************************************************
function dumptable(x)
    local result = '{'
    for i,v in ipairs(x) do
        if (type(v) == 'table') then
            result = result .. dumptable(v)
        else
            result = result .. v .. ','
        end
    end
    return result .. '}\n'
end
-- ------------
--print(dumptable(nodes))
--print(dumptable(elements))
--print(dumptable(materials))
--print(dumptable(allsections))
-- ***********************************************************************
-- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
model = StructureModel(2,3)
-- Assign all input data to Mercury
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(allsections)
model:addElements(elements)
-- ***********************************************************************
-- constrain bottom nodes
for bay=1,nbays+1 do
    model:constrainNode(columnnodename(bay,1,1),1,1,1)
end
-- ***********************************************************************
solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-5, iterations=100000})
-- ***********************************************************************
-- Monitoring node number
node06 = columnnodename(1,1,6);
node13 = columnnodename(1,2,7)
node20 = columnnodename(1,3,7)
node07 = columnnodename(1,1,7)
node14 = columnnodename(1,2,8)
node21 = columnnodename(1,3,8)
node48 = columnnodename(3,1,6)
node49 = columnnodename(3,1,7)
node01 = columnnodename(1,1,1)
node22 = columnnodename(2,1,1)
node43 = columnnodename(3,1,1)
node64 = columnnodename(4,1,1)
-- ***********************************************************************
disp_06_13_20 = {}
function disp1pertime(timestep)
  dx06,dy06,dz06 = model:nodeDisplacements(node06)
  dx13,dy13,dz13 = model:nodeDisplacements(node13)
  dx20,dy20,dz20 = model:nodeDisplacements(node20)
    table.insert(disp_06_13_20,dx06)
    table.insert(disp_06_13_20,dy06)
    table.insert(disp_06_13_20,dz06)
    table.insert(disp_06_13_20,dx13)
    table.insert(disp_06_13_20,dy13)
    table.insert(disp_06_13_20,dz13)
    table.insert(disp_06_13_20,dx20)
    table.insert(disp_06_13_20,dy20)
    table.insert(disp_06_13_20,dz20)
end
disp_07_14_21 = {}
function disp2pertime(timestep)
  dx07,dy07,dz07 = model:nodeDisplacements(node07)
  dx14,dy14,dz14 = model:nodeDisplacements(node14)
  dx21,dy21,dz21 = model:nodeDisplacements(node21)
    table.insert(disp_07_14_21,dx07)
    table.insert(disp_07_14_21,dy07)
    table.insert(disp_07_14_21,dz07)
    table.insert(disp_07_14_21,dx14)
    table.insert(disp_07_14_21,dy14)
    table.insert(disp_07_14_21,dz14)
    table.insert(disp_07_14_21,dx21)
    table.insert(disp_07_14_21,dy21)
    table.insert(disp_07_14_21,dz21)
end
disp_48_49 = {}
function disp3pertime(timestep)
  dx48,dy48,dz48 = model:nodeDisplacements(node48)
  dx49,dy49,dz49 = model:nodeDisplacements(node49)
    table.insert(disp_48_49,dx48)
    table.insert(disp_48_49,dy48)
    table.insert(disp_48_49,dz48)
    table.insert(disp_48_49,dx49)
    table.insert(disp_48_49,dy49)
    table.insert(disp_48_49,dz49)
end
--
react = {}
function reactpertime(timestep)
    fx01,fy01,fz01 = model:nodeRestoringForces(node01)
    fx22,fy22,fz22 = model:nodeRestoringForces(node22)
    fx43,fy43,fz43 = model:nodeRestoringForces(node43)
    fx64,fy64,fz64 = model:nodeRestoringForces(node64)
    table.insert(react, fx01)
    table.insert(react, fy01)
    table.insert(react, fz01)
    table.insert(react, fx22)
    table.insert(react, fy22)
    table.insert(react, fz22)
    table.insert(react, fx43)
    table.insert(react, fy43)
    table.insert(react, fz43)
    table.insert(react, fx64)
    table.insert(react, fy64)
    table.insert(react, fz64)
```

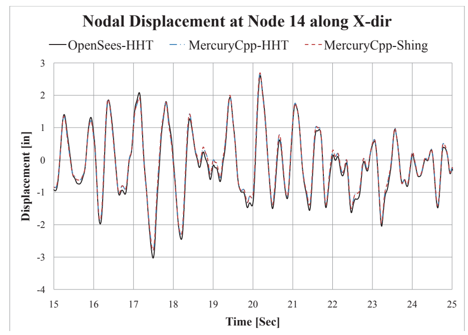
```
    print(timestep);
end
--
-- *********************************************************************
earthquakeloading = LoadDescription()
accelamp = 1.0*g
earthquakeloading:addLoad({'groundmotion',earthquakefile .. ',dt=0.005', 1,accelamp})
-- *********************************************
--transientanalysis = DynamicAnalysis("hht", model, solver, earthquakeloading,
 --[[ dt, alpha, beta, gamma]] 0.005, alpha, beta,gamma)
transientanalysis = DynamicAnalysis("hybridhht", model, solver, earthquakeloading,
                    --[[ dt, alpha, beta, gamma, iteration]] 0.005, alpha, beta,gamma, 10)
model:setRayleighCoefficients(1.1770500887303603,0.0015992014979696392)
-- *********************************************
transientanalysis:addcallback(disp1pertime, "timestep")
transientanalysis:addcallback(disp2pertime, "timestep")
transientanalysis:addcallback(disp3pertime, "timestep")
transientanalysis:addcallback(reactpertime, "timestep")
-- *********************************************
--transientanalysis:solve(15281)
transientanalysis:solve(152810)
-- *******************************************************
-- Set output file
function writedata6(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 5) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
-- ----------
function writedata9(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 8) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
-- ----------
function writedata12(x, fname)
    local f = assert(io.open(fname,'w'))
    local writenl = 0
    for i,v in ipairs(x) do
        f:write(v, " ")
        writenl = writenl + 1
        -- length of row size: writenl
        if (writenl > 11) then
            writenl = 0
            f:write("\n")
        end
    end
    f:close()
end
-- *******************************************************
--writedata9(disp_06_13_20,'Ex29HHT_06_13_20_displ.dat')
--writedata9(disp_07_14_21,'Ex29HHT_07_14_21_displ.dat')
--writedata6(disp_48_49,'Ex29HHT_48_49_displ.dat')
--writedata12(react,'Ex29HHT_01_22_43_64_react.dat')

writedata9(disp_06_13_20,'Ex29Shing_06_13_20_displ.dat')
writedata9(disp_07_14_21,'Ex29Shing_07_14_21_displ.dat')
writedata6(disp_48_49,'Ex29Shing_48_49_displ.dat')
writedata12(react,'Ex29Shing_01_22_43_64_react.dat')
-- *******************************************************
```

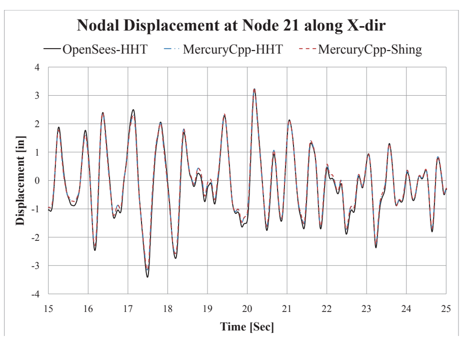
Fig. 17 shows the comparison for Mercury C++ and OpenSees.



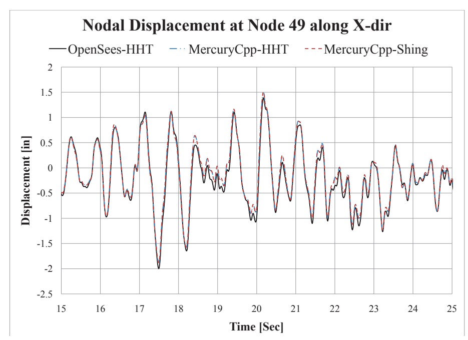
(a) Node 7



(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