The OpenSees Examples Primer

Version 1.2
August 20, 2001

Frank McKenna and Michael Scott
Pacific Earthquake Engineering Research Center
University of California, Berkeley

Introduction

The objective of this primer is to provide new users of OpenSees (Open System for Earthquake Engineering Simulation) familiar structural engineering examples as a convenient method for learning how to use the software. OpenSees is an object-oriented framework for building models of structural and geotechnical systems, performing nonlinear analysis with the model, and processing the response results. The goal for OpenSees is to support a wide range of simulation applications in earthquake engineering. The details, however, on how OpenSees accomplishes this goal are not particularly important for new users, who are primarily interested in how to solve problems.

This primer examines a few typical examples. Most users will conduct a simulation with a scripting language that has been extended to incorporate the features of OpenSees. As new features are developed, such as material models, elements, solution methods, etc., the scripting language can be extended to include them. The scripting language is named Tcl/Tk, and it has many features for dealing with variables, expressions, loops, data structures, input/output, that are useful for doing a simulation. Some of the basic features of Tcl will be illustrated in the examples.

Although users do not need to understand the object-oriented principles in the OpenSees framework, some terminology helps in the description of the examples. We talk about commands creating objects, which may be a specific material, element, analysis procedure, etc. To conduct a simulation, the user creates objects for three main purposes:

1. **Modeling**: The user first creates a ModelBuilder object which defines the type of model, and commands available for building the model. With a ModelBuilder defined, the user then creates the Element, Node, LoadPattern and Constraint objects that define the model. In this primer, the use of a basic ModelBuilder will be demonstrated.

2. **Analysis**: After defined the model, the next step is to create the Analysis object for analyzing the model. This may be a simple static linear analysis or a transient nonlinear analysis. In OpenSees, an Analysis object is composed of several component objects, which define how the analysis is performed. The component objects consist of the following: SolutionAlgorithm, Integrator, ConstraintHandler, DOF_Numberer, SystemOfEqn, Solver, and AnalysisModel. This approach provides a great deal of flexibility in how an analysis is conducted.

3. **Output Specification**: Once the model and analysis have been defined, the user has the option of specifying what is to be monitored during the analysis. This, for example, could be the displacement history at a node or internal state of an element in a transient analysis or the entire state of the model at each step in the solution procedure. Several Recorder objects are created to store what the user wants to examine.

In the examples, Tcl scripts are used to create a model, analysis, and output specification. The examples are (1) simple truss structure, (2) reinforced concrete portal frame, (3) two-story multi-bay reinforced concrete frame, and (4) a three-dimensional frame. The examples are not meant to be completely realistic, but they are representative of typical structures. The analyses performed on these models consist of simple static analysis, pushover analysis and transient analysis. An example of moment-curvature analysis is also performed on a reinforced concrete section.

1 EXAMPLE 1 - Truss Example

The first example is a simple truss structure. The purpose of this example is to show that model generation in OpenSees can resemble typical finite element analysis programs with the definition of nodes, materials, elements, loads and constraints. The example also demonstrates how an analysis object is 'built' from component objects.

1.1 Example 1.1

This example is of a linear-elastic three bar truss, as shown in figure 1, subject to static loads.

Files Required

1. Example1.1.tcl

Model

The model consists of four nodes, three truss elements, a single load pattern with a nodal load acting at node 4, and constraints at the three support nodes. Since the truss elements have the same elastic material, a single Elastic material object is created.

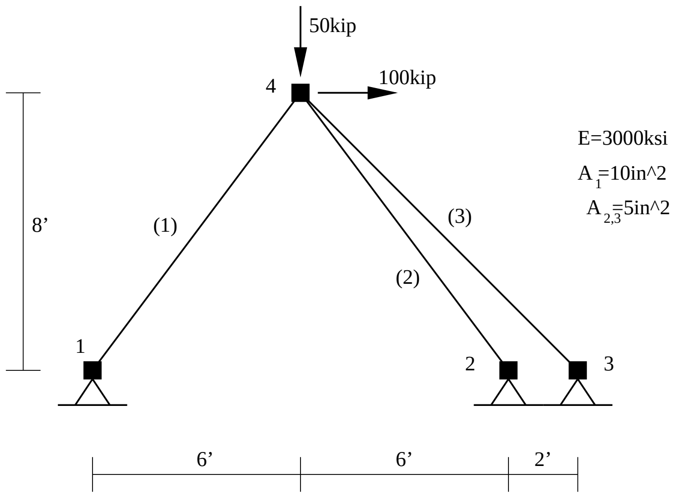

Figure 1: Example 1.1

Analysis

The model is linear, so we use a solution Algorithm of type Linear. Even though the solution is linear, we have to select a procedure for applying the load, which is called an Integrator. For this problem, a LoadControl integrator advances the solution. The equations are formed using a banded system, so the System is BandSPD (banded, symmetric positive definite). This is a good choice for most moderate size models. The equations have to be numbered, so typically an RCM numberer object is used (for Reverse Cuthill-McKee). The constraints are most easily represented with a Plain constraint handler.

Once all the components of an analysis are defined, the Analysis object itself is created. For this problem a Static Analysis object is used.

Output Specification

When the analysis is complete the state of node 4 and all three elements will be printed to the screen. Nothing is recorded for later use.

OpenSees Script

The Tcl script for the example is shown below. A comment is indicated by a # symbol. In the comments below, the syntax for important commands are given.

```
# OpenSees Example 1.1
# OpenSees Primer
#
# Units: kips, in, sec

# ------------------------------
# Start of model generation
# ------------------------------

# Create ModelBuilder (with two-dimensions and 2 DOF/node)
model BasicBuilder -ndm 2 -ndf 2

# Create nodes
# ------------

# Create nodes - command: node nodeId xCrd yCrd
node 1   0.0  0.0
node 2 144.0  0.0
node 3 168.0  0.0
node 4  72.0 96.0

# Set the boundary conditions - command: fix nodeID xFix? yFix?
fix 1 1 1
fix 2 1 1
fix 3 1 1

# Define materials for truss elements
# -----------------------------------

# Create Elastic material - command: uniaxialMaterial Elastic matID E
uniaxialMaterial Elastic 1 3000
```

```
# Define elements
# ---------------

# Create truss elements - command: element truss trussID node1 node2 A matID
element truss 1 1 4 10.0 1
element truss 2 2 4 5.0  1
element truss 3 3 4 5.0  1

# Define loads
# ------------

# Create a Plain load pattern with a linear TimeSeries
pattern Plain 1 "Linear" {

    # Create the nodal load - command: load nodeID xForce yForce
    load 4 100 -50
}

# ------------------------------
# End of model generation
# ------------------------------

# ------------------------------
# Start of analysis generation
# ------------------------------

# Create the solution algorithm, a Linear algorithm is created
algorithm Linear

# Create the integration scheme, LoadControl using steps of 1.0
integrator LoadControl 1.0 1 1.0 1.0

# Create the system of equation, a SPD using a band storage scheme
system BandSPD

# Create the DOF numberer, the reverse Cuthill-McKee algorithm
numberer RCM

# Create the constraint handler, a Plain handler for homogeneous constraints
constraints Plain

# create the analysis object
analysis Static

# ------------------------------
# End of analysis generation
# ------------------------------
# ------------------------------
# Finally perform the analysis
# ------------------------------

# Perform the analysis for 1 load step
analyze 1

# Print the current state at node 4 and at all elements
print node 4
print element
```

Results

```
 Node: 4
        Coordinates  : 72 96
        commitDisps: 0.530093 -0.177894
        unbalanced Load: 100 -50

Element: 1 type: Truss  iNode: 1 jNode: 4 Area: 10 Total Mass: 0
         strain: 0.00146451 axial load: 43.9352
         unbalanced load: -26.3611 -35.1482 26.3611 35.1482
         Material: Elastic tag: 1
         E: 3000 eta: 0

Element: 2 type: Truss  iNode: 2 jNode: 4 Area: 5 Total Mass: 0
         strain: -0.00383642 axial load: -57.5463
         unbalanced load: -34.5278 46.0371 34.5278 -46.0371
         Material: Elastic tag: 1
         E: 3000 eta: 0

Element: 3 type: Truss  iNode: 3 jNode: 4 Area: 5 Total Mass: 0
         strain: -0.00368743 axial load: -55.3114
         unbalanced load: -39.1111 39.1111 39.1111 -39.1111
         Material: Elastic tag: 1
         E: 3000 eta: 0
```

For the node, displacements and loads are given. For the truss elements, the axial strain and force are provided along with the resisting forces in the global coordinate system.

2 EXAMPLE 2 - Moment-Curvature Analysis of a Reinforced Concrete Section

This next example covers the moment-curvature analysis of a reinforced concrete section. The zero-length element with a fiber discretization of the cross section is used in the model. In addition, Tcl language features such as variable and command substitution, expression evaluation, and procedures are demonstrated.

2.1 Example 2.1

In this example, a moment-curvature analysis of the fiber section is undertaken. Figure 4 shows the fiber discretization for the section.

Files Required

1. Example2.1..tcl
2. MomentCurvature.tcl

Model

The model consists of two nodes and a ZeroLengthSection element. A depiction of the element geometry is shown in figure 2. The drawing on the left of figure 2 shows an edge view of the element where the local z-axis, as seen on the right side of the figure and in figure 3, is coming out of the page. Node 1 is completely restrained, while the applied loads act on node 2. A compressive axial load, P, of 180 kips is applied to the section during the moment-curvature analysis.

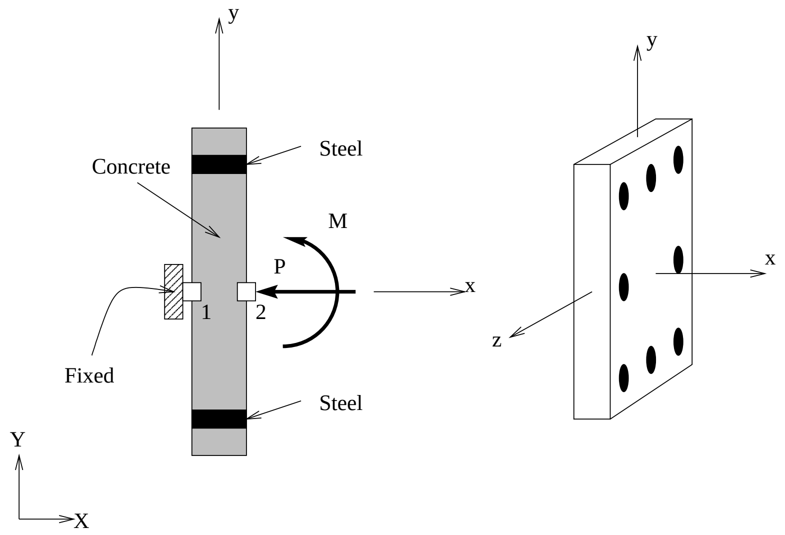

Figure 2: Geometry of zero-length element

For the zero length element, a section discretized by concrete and steel is created to represent the resultant behavior. UniaxialMaterial objects are created to define the fiber stress-strain relationships: confined concrete in the column core, unconfined concrete in the column cover, and reinforcing steel.

The dimensions of the fiber section are shown in figure 3. The section depth is 24 inches, the width is 15 inches, and there are 1.5 inches of cover around the entire section. Strong axis bending is about the section z-axis. In fact, the section z-axis is the strong axis of bending for all fiber sections in planar problems. The section is separated into confined and unconfined concrete regions, for which separate fiber discretizations will be generated. Reinforcing steel bars will be placed around the boundary of the confined and unconfined regions. The fiber discretization for the section is shown in figure 4.

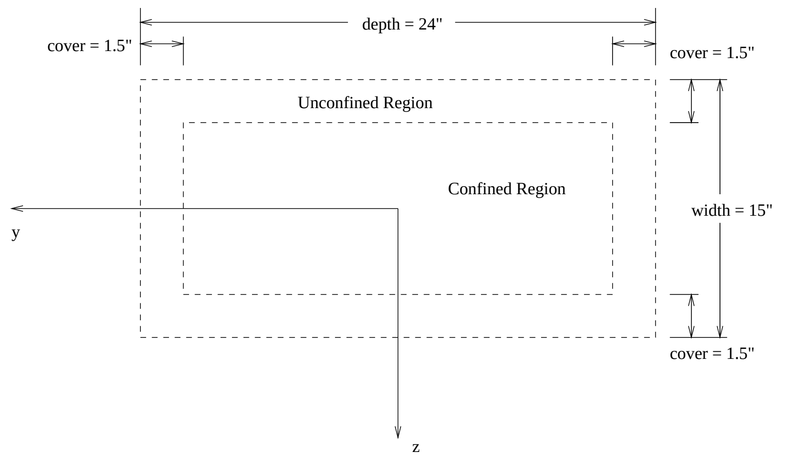

Figure 3: Dimensions of RC section

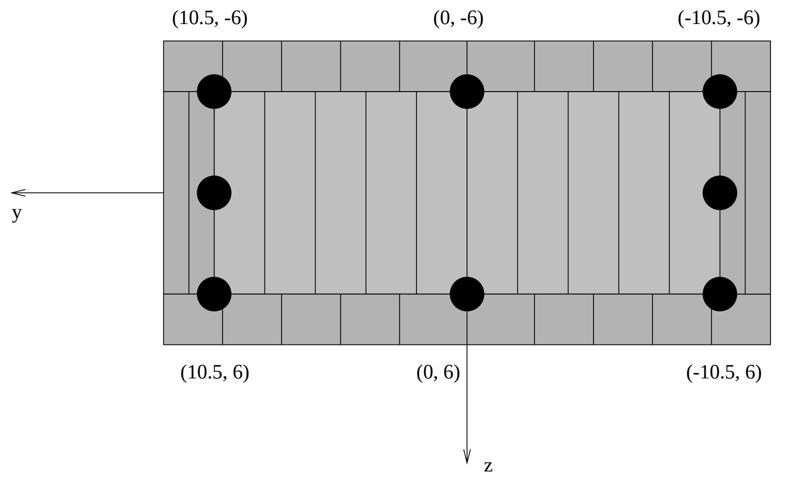

Figure 4: Fiber section discretization

Analysis

The section analysis is performed by the Tcl procedure MomentCurvature defined in the file MomentCurvature.tcl. The arguments to the procedure are the tag of the section to be analyzed, the axial load applied to the section, the maximum curvature, and the number of displacement increments to reach the maximum curvature.

Output Specification

The output for the moment-curvature analysis will be the section forces and deformations, stored in the file section1.out. In addition, an estimate of the section yield curvature is printed to the screen.

OpenSees Script

In the script below variables, are set and can then be used with the syntax of $variable. Expressions can be evaluated, although the Tcl syntax at first appears cumbersome. An expression is given by an expr command enclosed in square brackets []'s. Typically, the result of an expression is then set to another variable. A simple example to add 2.0 to a parameter is shown below:

```
set v 3.0
set sum [expr $v + 2.0]
puts $sum;    # print the sum
```

Comments with # can appear on the same line as a command, but then the command must be terminated with a semi-colon.

```
# OpenSees Example 2.1
# OpenSees Primer
#
# Units: kips, in, sec

# Define model builder
# --------------------
model BasicBuilder -ndm 2 -ndf 3

# Define materials for nonlinear columns
# ------------------------------------------
# CONCRETE                  tag   f'c        ec0   f'cu        ecu
# Core concrete (confined)
uniaxialMaterial Concrete01  1  -6.0  -0.004   -5.0     -0.014

# Cover concrete (unconfined)
uniaxialMaterial Concrete01  2  -5.0   -0.002   0.0     -0.006

# STEEL
# Reinforcing steel
set fy 60.0;      # Yield stress
set E 30000.0;    # Young's modulus
#                         tag  fy E0    b
uniaxialMaterial Steel01  3  $fy $E 0.01

# Define cross-section for nonlinear columns
# ------------------------------------------

# set some parameters
set colWidth 15
set colDepth 24

set cover  1.5
set As    0.60;     # area of no. 7 bars

# some variables derived from the parameters
set y1 [expr $colDepth/2.0]
set z1 [expr $colWidth/2.0]

section Fiber 1 {

    # Create the concrete core fibers
    patch rect 1 10 1 [expr $cover-$y1] [expr $cover-$z1]
                      [expr $y1-$cover] [expr $z1-$cover]

    # Create the concrete cover fibers (top, bottom, left, right)
    patch rect 2 10 1  [expr -$y1] [expr $z1-$cover]
                               $y1                     $z1
    patch rect 2 10 1  [expr -$y1]             [expr -$z1]
                               $y1       [expr $cover-$z1]
    patch rect 2  2 1  [expr -$y1]       [expr $cover-$z1]
                       [expr $cover-$y1] [expr $z1-$cover]
    patch rect 2  2 1  [expr $y1-$cover] [expr $cover-$z1]
                                     $y1 [expr $z1-$cover]

    # Create the reinforcing fibers (left, middle, right)
    layer straight 3 3 $As [expr $y1-$cover] [expr $z1-$cover]
                           [expr $y1-$cover] [expr $cover-$z1]
    layer straight 3 2 $As 0.0 [expr $z1-$cover]
                           0.0 [expr $cover-$z1]
    layer straight 3 3 $As [expr $cover-$y1] [expr $z1-$cover]
                           [expr $cover-$y1] [expr $cover-$z1]

}

# Estimate yield curvature
# (Assuming no axial load and only top and bottom steel)
set d [expr $colDepth-$cover] ;# d -- from cover to rebar
set epsy [expr $fy/$E] ;# steel yield strain
set Ky [expr $epsy/(0.7*$d)]

# Print estimate to standard output
puts "Estimated yield curvature: $Ky"

# Set axial load
set P -180

set mu 15; # Target ductility for analysis
set numIncr 100; # Number of analysis increments

# Call the section analysis procedure
source MomentCurvature.tcl
MomentCurvature 1 $P [expr $Ky*$mu] $numIncr
```

The Tcl procedure to perform the moment-curvature analysis follows. In this procedure, the nodes are defined to be at the same geometric location and the ZeroLengthSection element is used. A single load step is performed for the axial load, then the integrator is changed to DisplacementControl to impose nodal displacements, which map directly to section deformations. A reference moment of 1.0 is defined in a Linear time series. For this reference moment, the DisplacementControl integrator will determine the load factor needed to apply the imposed displacement. A node recorder is defined to track the moment-curvature results. The load factor is the moment, and the nodal rotation is in fact the curvature of the element with zero thickness.

```
# Arguments
# secTag -- tag identifying section to be analyzed
# axialLoad -- axial load applied to section (negative is compression)
# maxK -- maximum curvature reached during analysis
# numIncr -- number of increments used to reach maxK (default 100)
#
# Sets up a recorder which writes moment-curvature results to file
# section$secTag.out ... the moment is in column 1, and curvature in column 2

proc MomentCurvature {secTag axialLoad maxK {numIncr 100} } {
   # Define two nodes at (0,0)
   node 1 0.0 0.0
   node 2 0.0 0.0

   # Fix all degrees of freedom except axial and bending at node 2
   fix 1 1 1 1
   fix 2 0 1 0

   # Define element
   #                         tag ndI ndJ  secTag
   element zeroLengthSection  1   1   2  $secTag
   # Create recorder
   recorder Node section$secTag.out disp -time -node 2 -dof 3

   # Define constant axial load
   pattern Plain 1 "Constant" {
      load 2 $axialLoad 0.0 0.0
   }

   # Define analysis parameters
   integrator LoadControl 0 1 0 0
   system SparseGeneral -piv;
   test NormUnbalance 1.0e-9 10
   numberer Plain
   constraints Plain
   algorithm Newton
   analysis Static

   # Do one analysis for constant axial load
   analyze 1

   # Define reference moment
   pattern Plain 2 "Linear" {
      load 2 0.0 0.0 1.0
   }

   # Compute curvature increment
   set dK [expr $maxK/$numIncr]

   # Use displacement control at node 2 for section analysis
   integrator DisplacementControl 2 3 $dK 1 $dK $dK

   # Do the section analysis
   analyze $numIncr
}
```

Results

```
Estimated yield curvature: 0.000126984126984
```

The file section1.out contains for each committed state a line with the load factor and the rotation at node 3. This can be used to plot the moment-curvature relationship as shown in figure 5.

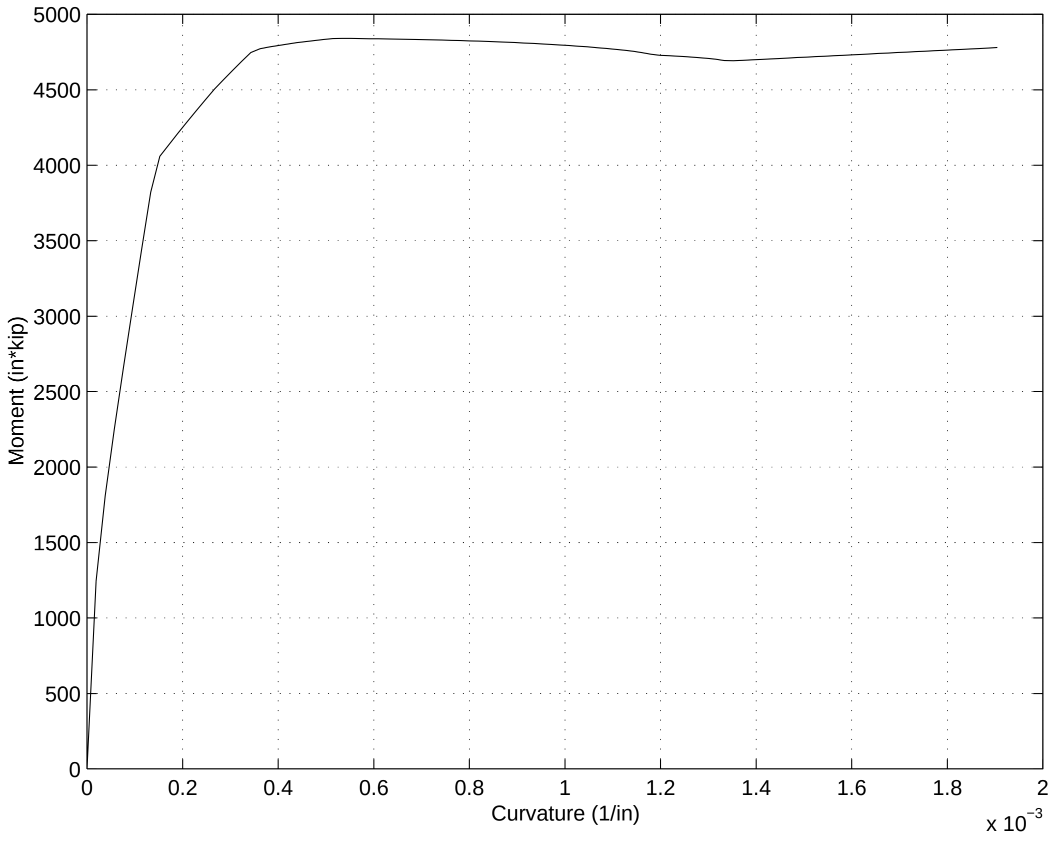

Figure 5: Moment-curvature analysis of column section

3 EXAMPLE 3 - Portal Frame Examples

This next set of examples covers the nonlinear analysis of a reinforced concrete frame. The nonlinear beam column element with a fiber discretization of the cross section is used in the model. In addition, Tcl language features such as variable and command substitution, expression evaluation, the if-then-else control structure, and procedures are demonstrated in several elaborations of the example.

3.1 Example 3.1

This example is of a reinforced concrete portal frame, as shown in figure 6, subject to gravity loads.

Files Required

1. Example3.1.tcl

Model

A nonlinear model of the portal frame shown in figure 6 is created. The model consists of four nodes, two nonlinear beam-column elements, 1 and 2, to model the columns and an elastic beam element, 3, to model the beam. For the column elements a section, identical to the section used in Example 2, is created using steel and concrete fibers.

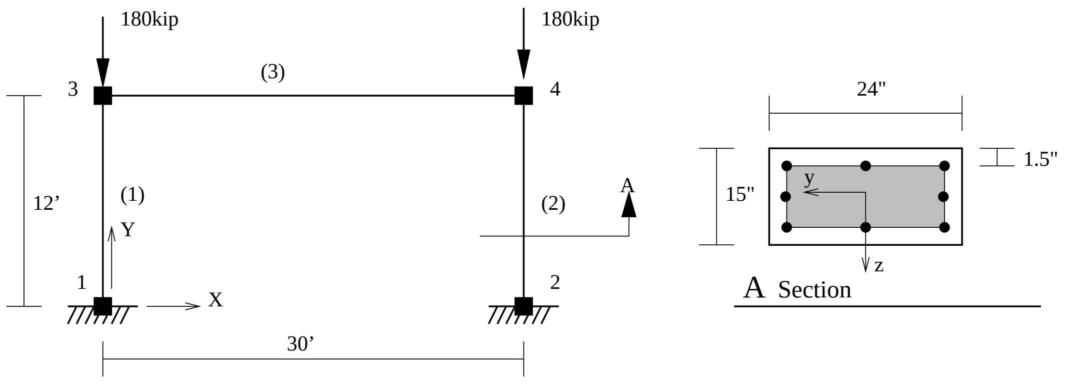

Figure 6: Example 3.1

A single load pattern with a linear time series, two vertical nodal loads acting at nodes 3 and 4, and single point constraints to constrain nodes 1 and 2 are created.

Analysis

The model contains material non-linearities, so a solution Algorithm of type Newton is used. The solution algorithm uses a ConvergenceTest which tests convergence of the equilibrium solution with the norm of the displacement increment vector. For this nonlinear problem, the gravity loads are applied incrementally until the full load is applied. To achieve this, a LoadControl integrator which advances the solution with an increment of 0.1 at each load step is used. The equations are formed using a banded storage scheme, so the System is BandGeneral. The equations are numbered using an RCM (reverse Cuthill-McKee) numberer. The constraints are enforced with a Plain constraint handler.

Once all the components of an analysis are defined, the Analysis object itself is created. For this problem a Static Analysis object is used. To achieve the full gravity load, 10 load steps are performed.

Output Specification

At end of analysis, the state at nodes 3 and 4 is output. The state of element 1 is also output.

OpenSees Script

```
# OpenSees Example 3.1
# OpenSees Primer
#
# Units: kips, in, sec

# ------------------------------
# Start of model generation
# ------------------------------

# Create ModelBuilder (with two-dimensions and 3 DOF/node)
model basic -ndm 2 -ndf 3

# Create nodes
# ------------

# Set parameters for overall model geometry
set width    360
set height   144

# Create nodes
#    tag        X       Y
node  1       0.0     0.0
node  2    $width     0.0
node  3       0.0 $height
node  4    $width $height

# Fix supports at base of columns
#    tag   DX   DY   RZ
fix   1     1    1    1
fix   2     1    1    1

# Define materials for nonlinear columns
# ------------------------------------------
# CONCRETE                  tag   f'c        ec0   f'cu        ecu
# Core concrete (confined)
uniaxialMaterial Concrete01  1  -6.0  -0.004   -5.0     -0.014

# Cover concrete (unconfined)
uniaxialMaterial Concrete01  2  -5.0   -0.002   0.0     -0.006

# STEEL
# Reinforcing steel
set fy 60.0;      # Yield stress
set E 30000.0;    # Young's modulus
#                        tag  fy E0    b
uniaxialMaterial Steel01  3  $fy $E 0.01

# Define cross-section for nonlinear columns
# ------------------------------------------

# set some parameters
set colWidth 15
set colDepth 24

set cover  1.5
set As    0.60;     # area of no. 7 bars

# some variables derived from the parameters
set y1 [expr $colDepth/2.0]
set z1 [expr $colWidth/2.0]

section Fiber 1 {

    # Create the concrete core fibers
    patch rect 1 10 1 [expr $cover-$y1] [expr $cover-$z1]
                      [expr $y1-$cover] [expr $z1-$cover]

    # Create the concrete cover fibers (top, bottom, left, right)
    patch rect 2 10 1  [expr -$y1] [expr $z1-$cover]
                               $y1               $z1
    patch rect 2 10 1  [expr -$y1]       [expr -$z1]
                               $y1 [expr $cover-$z1]
    patch rect 2  2 1  [expr -$y1] [expr $cover-$z1]
                       [expr $cover-$y1] [expr $z1-$cover]
    patch rect 2  2 1  [expr $y1-$cover] [expr $cover-$z1]
                                     $y1 [expr $z1-$cover]

    # Create the reinforcing fibers (left, middle, right)
    layer straight 3 3 $As [expr $y1-$cover] [expr $z1-$cover]
                           [expr $y1-$cover] [expr $cover-$z1]
    layer straight 3 2 $As              0.0 [expr $z1-$cover]
                                        0.0 [expr $cover-$z1]
    layer straight 3 3 $As [expr $cover-$y1] [expr $z1-$cover]
                           [expr $cover-$y1] [expr $cover-$z1]

}

# Define column elements
# ----------------------

# Geometry of column elements
#               tag
geomTransf Linear 1

# Number of integration points along length of element
set np 5

# Create the coulumns using Beam-column elements
#                           tag ndI ndJ nsecs secID transfTag
element nonlinearBeamColumn  1   1   3   $np    1       1
element nonlinearBeamColumn  2   2   4   $np    1       1

# Define beam elment
# -----------------------------

# Geometry of column elements
#               tag
geomTransf Linear 2

# Create the beam element
#                          tag ndI ndJ     A       E    Iz   transfTag
element elasticBeamColumn   3   3   4    360    4030  8640    2

# Define gravity loads
# --------------------

# Set some parameter
set P 180;                # 10% of axial capacity of columns

# Create a Plain load pattern with a Linear TimeSeries
pattern Plain 1 "Linear" {

        # Create nodal loads at nodes 3 & 4
        #    nd    FX          FY  MZ
        load  3   0.0  [expr -$P] 0.0
        load  4   0.0  [expr -$P] 0.0
}
```

```
# ------------------------------
# End of model generation
# ------------------------------


# ------------------------------
# Start of analysis generation
# ------------------------------

# Create the system of equation, a banded general system
system BandGeneral

# Create the constraint handler, the transformation method
constraints Transformation

# Create the DOF numberer, a plain numbering scheme
numberer RCM

# Create the convergence test, the norm of the residual with a tolerance of
# 1e-12 and a max number of iterations of 10
test NormDispIncr 1.0e-12  10 3

# Create the solution algorithm, a Newton-Raphson algorithm
algorithm Newton

# Create the integration scheme, the LoadControl scheme using steps of 0.1
integrator LoadControl 0.1 1 0.1 0.1

# Create the analysis object
analysis Static

# ------------------------------
# End of analysis generation
# ------------------------------


# ------------------------------
# Finally perform the analysis
# ------------------------------

# initialize in case we need to do an initial stiffness iteration
initialize

# Perform the gravity load analysis, requires 10 steps to reach the load level
analyze 10
```

```
# Print out the state of nodes 3 and 4
print node 3 4

# Print out the state of element 1
print element 1
```

Results

```
 Node: 3
        Coordinates  : 0 144
        commitDisps: -4.10875e-18 -0.0183736 4.97076e-20
        unbalanced Load: 0 -180 0

 Node: 4
        Coordinates  : 360 144
        commitDisps: -4.10842e-18 -0.0183736 4.92006e-20
        unbalanced Load: 0 -180 0

 Element: 1 Type: NLBeamColumn2d         Connected Nodes: 1 3
        Number of Sections: 5   Mass density: 0
        End 1 Forces (P V M): 180 -1.75302e-31 4.9738e-14
        End 2 Forces (P V M): -180 1.75302e-31 -4.9738e-14
        Resisting Force: 1.47911e-31 180 4.9738e-14 -1.47911e-31 -180 -4.9738e-14
```

For the two nodes, displacements and loads are given. For the nonlinear beam-column element, the element end forces in the local system are provided along with the resisting forces in the global coordinate system.

3.2 Example 3.2

In this example the nonlinear reinforced concrete portal frame which has undergone the gravity load analysis of Example 3.1 is now subjected to a pushover analysis.

Files Required

1. Example3.2.tcl
2. Example3.1.tcl

Model

After performing the gravity load analysis on the model, the time in the domain is reset to 0.0 and the current value of all loads acting are held constant. A new load pattern with a linear time series and horizontal loads acting at nodes 3 and 4 is then added to the model.

Analysis

The static analysis used to perform the gravity load analysis is modified to take a new DisplacementControl integrator. At each new step in the analysis the integrator will determine the load increment necessary to increment the horizontal displacement at node 3 by 0.1 in. 60 analysis steps are performed in this new analysis.

Output Specification

For this analysis the nodal displacements at nodes 3 and 4 will be stored in the file nodePushover.out for post-processing. In addition, the end forces in the local coordinate system for elements 1 and 2 will be stored in the file elePushover.out. At the end of the analysis, the state of node 3 is printed to the screen.

OpenSees Script

```
# OpenSees Example 3.2
# OpenSees Primer
#
# Units: kips, in, sec

# ----------------------------------------------------
# Start of Model Generation & Initial Gravity Analysis
# ----------------------------------------------------

# Do operations of Example3.1 by sourcing in the tcl file
source Example3.1.tcl
puts ``Gravity load analysis completed''

# Set the gravity loads to be constant & reset the time in the domain
loadConst -time 0.0
```

```
# ----------------------------------------------------
# End of Model Generation & Initial Gravity Analysis
# ----------------------------------------------------


# ----------------------------------------------------
# Start of additional modeling for lateral loads
# ----------------------------------------------------

# Define lateral loads
# --------------------

# Set some parameters
set H 10.0; # Reference lateral load

# Set lateral load pattern with a Linear TimeSeries
pattern Plain 2 "Linear" {

        # Create nodal loads at nodes 3 & 4
        #    nd    FX  FY  MZ
        load 3 $H 0.0 0.0
        load 4 $H 0.0 0.0
}

# ----------------------------------------------------
# End of additional modeling for lateral loads
# ----------------------------------------------------


# ----------------------------------------------------
# Start of modifications to analysis for push over
# ----------------------------------------------------

# Set some parameters
set dU 0.1;         # Displacement increment

# Change the integration scheme to be displacement control
#                             node dof init Jd min max
integrator DisplacementControl  3   1   $dU  1 $dU $dU

# ----------------------------------------------------
# End of modifications to analysis for push over
# ----------------------------------------------------


# ------------------------------
# Start of recorder generation
# ------------------------------

# Create a recorder to monitor nodal displacements
recorder Node node32.out disp -time -node 3 4 -dof 1 2 3

# Create a recorder to monitor element forces in columns
recorder Element 1 2 -time -file ele32.out localForce

# --------------------------------
# End of recorder generation
# ---------------------------------


# ------------------------------
# Finally perform the analysis
# ------------------------------

# Set some parameters
set maxU 6.0;         # Max displacement
set numSteps [expr int($maxU/$dU)]

# Perform the analysis
analyze $numSteps
puts ``Pushover analysis completed''

# Print the state at node 3
print node 3
```

Results

```
Gravity load analysis completed
Setting time in domain to be : 0.0

Pushover analysis completed
 Node: 3
        Coordinates  : 0 144
        commitDisps: 6 0.488625 -0.00851377
        unbalanced Load: 71.8819 -180 0
```

In addition to what is displayed on the screen, the file node32.out and ele32.out have been created by the script. Each line of node32.out contains the time, DX, DY and RZ for node 3 and DX, DY and RZ for node 4 at the end of an iteration. Each line of eleForce.out contains the time, and the element end forces in the local coordinate system. A plot of the load-displacement relationship at node 3 is shown in figure 7.

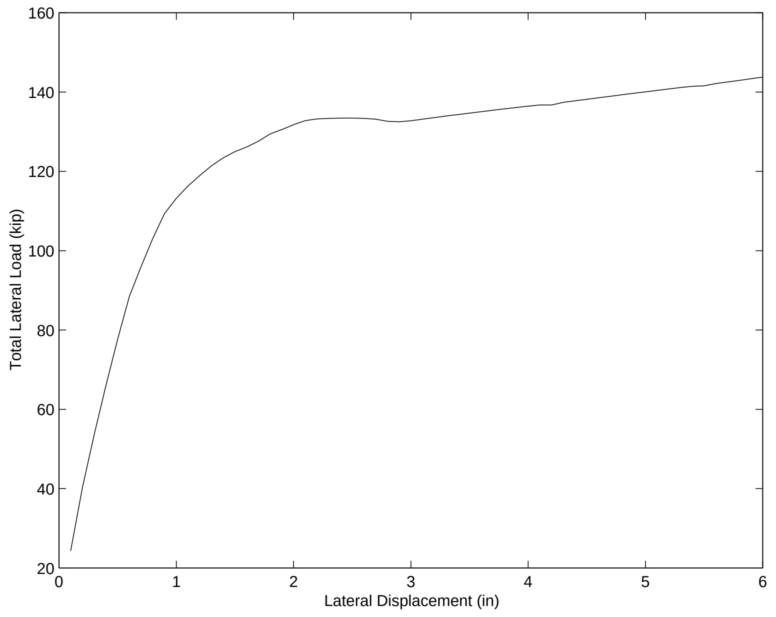

Figure 7: Load displacement curve for node 3

3.3 Example 3.3

In this example the reinforced concrete portal frame which has undergone the gravity load analysis of Example 3.1 is now subjected to a uniform earthquake excitation.

Files Required

1. Example3.3.tcl
2. Example3.1.tcl
3. ReadSMDFile.tcl

Model

After performing the gravity load analysis, the time in the domain is reset to 0.0 and the current value of all active loads is set to constant. Mass terms are added to nodes 3 and 4. A new uniform excitation load pattern is created. The excitation acts in the horizontal direction and reads the acceleration record and time interval from the file ARL360.g3. The file ARL360.g3 is created from the PEER Strong Motion Database (http://peer.berkeley.edu/smcat/) record ARL360.at2 using the Tcl procedure ReadSMDFile contained in the file ReadSMDFile.tcl.

Analysis

The static analysis object and its components are first deleted so that a new transient analysis object can be created.

A new solution Algorithm of type Newton is then created. The solution algorithm uses a ConvergenceTest which tests convergence on the norm of the displacement increment vector. The integrator for this analysis will be of type Newmark with a γ of 0.25 and a β of 0.5. The integrator will add some stiffness proportional damping to the system, the damping term will be based on the last committed stifness of the elements, i.e. $C = a_c K_{commit}$ with $a_c = 0.000625$. The equations are formed using a banded storage scheme, so the System is BandGeneral. The equations are numbered using an RCM (reverse Cuthill-McKee) numberer. The constraints are enforced with a Plain constraint handler.

Once all the components of an analysis are defined, the Analysis object itself is created. For this problem a Transient Analysis object is used. 2000 time steps are performed with a time step of 0.01.

In addition to the transient analysis, two eigenvalue analysis are performed on the model. The first is performed after the gravity analysis and the second after the transient analysis.

Output Specification

For this analysis the nodal displacenments at Nodes 3 and 4 will be stored in the file nodeTransient.out for post-processing. In addition the section forces and deformations for the section at the base of column 1 will also be stored in two seperate files. The results of the eigenvalue analysis will be displayed on the screen.

OpenSees Script

```
# OpenSees Example 3.3
# OpenSees Primer
#
# Units: kips, in, sec

# ----------------------------------------------------
# Start of Model Generation & Initial Gravity Analysis
# ----------------------------------------------------

# Do operations of Example3.1 by sourcing in the tcl file
source Example3.1.tcl
puts ‘‘Gravity load analysis completed’’

# Set the gravity loads to be constant & reset the time in the domain
loadConst -time 0.0

# ----------------------------------------------------
# End of Model Generation & Initial Gravity Analysis
# ----------------------------------------------------

# ----------------------------------------------------
# Start of additional modeling for dynamic loads
# ----------------------------------------------------

# Define nodal mass in terms of axial load on columns
set g 386.4
set m [expr $P/$g];       # expr command to evaluate an expression

#    tag   MX   MY   RZ
mass  3    $m   $m    0
mass  4    $m   $m    0

# Define dynamic loads
# --------------------

# Set some parameters
set outFile ARL360.g3
set accelSeries "Path -filePath $outFile -dt $dt -factor $g"

# Source in TCL proc to read a PEER Strong Motion Database record
source ReadSMDFile.tcl

# Perform the conversion from SMD record to OpenSees record and obtain dt
#              inFile     outFile dt
ReadSMDFile ARL360.at2   $outFile dt

# Create UniformExcitation load pattern
#                          tag dir
pattern UniformExcitation  2   1  -accel $accelSeries

# ---------------------------------------------------
# End of additional modeling for dynamic loads
# ---------------------------------------------------

# --------------------------------------------------------
# Start of modifications to analysis for transient analysis
# --------------------------------------------------------

# Delete the old analysis and all its component objects
wipeAnalysis

# Create the convergence test, the norm of the residual with a tolerance of
# 1e-12 and a max number of iterations of 10
test NormDispIncr 1.0e-12  10

# Create the solution algorithm, a Newton-Raphson algorithm
algorithm Newton

# Create the integration scheme, Newmark with gamma = 0.5 and beta =  0.25
integrator Newmark  0.5  0.25 0.0 0.0 0.0 0.000625

# Create the system of equation, a banded general storage scheme
system BandGeneral

# Create the constraint handler, a plain handler as homogeneous boundary conditions
constraints Plain

# Create the DOF numberer, the reverse Cuthill-McKee algorithm
numberer RCM

# Create the analysis object
analysis Transient

# --------------------------------------------------------
# End of modifications to analysis for transient analysis
# --------------------------------------------------------

# ------------------------------
# Start of recorder generation
# ------------------------------

# Create a recorder to monitor nodal displacements
recorder Node nodeTransient.out disp -time -node 3 -dof 1 2 3
```

```
# Create recorders to monitor section forces and deformations
# at the base of the left column
recorder Element 1 -time -file ele1secForce.out section 1 force
recorder Element 1 -time -file ele1secDef.out   section 1 deformation

# --------------------------------
# End of recorder generation
# ---------------------------------

# ------------------------------
# Finally perform the analysis
# ------------------------------

# Perform an eigenvalue analysis
puts [eigen 2]

# Perform the transient analysis
#         N   dt
set ok [analyze 2000 0.01]

# if the analysis fails at a step we try an
# initial stiffness iteration for that step
# and then go back to the current stifness for the rest

if {$ok != 0} {
    set tFinal [expr 2000 * 0.01]
    set tCurrent [getTime]
    set ok 0
    while {$ok == 0 && $tCurrent < $tFinal} {

        set ok [analyze 1 .01]

        # if the analysis fails try initial tangent iteration
        if {$ok != 0} {
            puts "regular newton failed .. lets try an initail stiffness for this step"
            test NormDispIncr 1.0e-12  100 1
            algorithm Newton -initial
            set ok [analyze 1 .01]
            if {$ok == 0} {puts "that worked .. back to regular newton"}
            test NormDispIncr 1.0e-12  10
            algorithm Newton
        }

        set tCurrent [getTime]

    }
}
```

```
if {$ok == 0} {
   puts "Transient analysis completed succesfully";
} else {
   puts "Transient analysis completed failed";
}

# Perform an eigenvalue analysis
puts [eigen 2]

# Print state of node 3
print node 3
```

Results

```
Gravity load analysis completed
Setting time in domain to be : 0.0
Eigenvalues: 269.542 17507.1

regular newton failed .. lets try an initail stiffness for this step

that worked .. back to regular newton

Transient analysis completed succesfully
1.669721e+02  1.734707e+04

 Node: 3
Coordinates  : 0 144
commitDisps: -0.101087 -0.0220163 0.000563238
Velocities   : -1.5214 0.022985 0.00963654
commitAccels: 4.11459 0.263923 -97.8123
 unbalanced Load: -3.9475 -180 0
Mass :
0.465839 0 0
0 0.465839 0
0 0 0

 Eigenvectors:
-1.03582 -0.979573
0.0130843 0.299073
0.00666293 0.00498144
```

The two eigenvalues for the eigenvalue analysis are printed to the screen. The state of node 3 at the end of the analysis is also printed. The information contains the last committed displacements, velocities and accelerations at the node, the unbalanced nodal forces and the nodal masses. In addition, the eigenvector components of the eigenvector pertaining to the node 3 is also displayed.

In addition to the contents displayed on the screen, three files have been created. Each line of nodeTransient.out contains the domain time, and DX, DY and RZ for node 3. Plotting the first and second columns of this file the lateral displacement versus time for node 3 can be obtained as shown in figure 8. Each line of the files ele1secForce.out and ele1secDef.out contain the domain time and the forces and deformations for section 1 (the base section) of element 1. These can be used to generate the moment-curvature time history of the base section of column 1 as shown in figure 9.

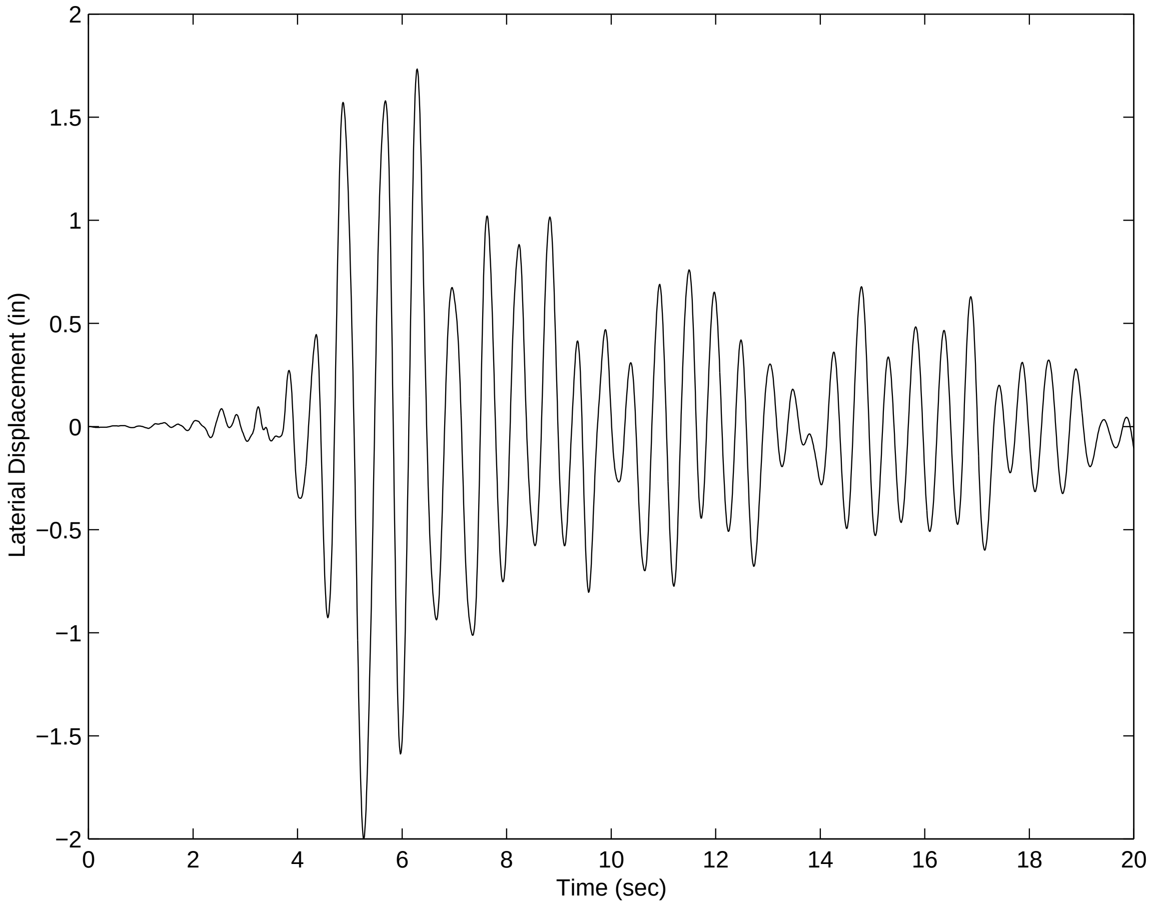

Figure 8: Lateral displacement at node 3

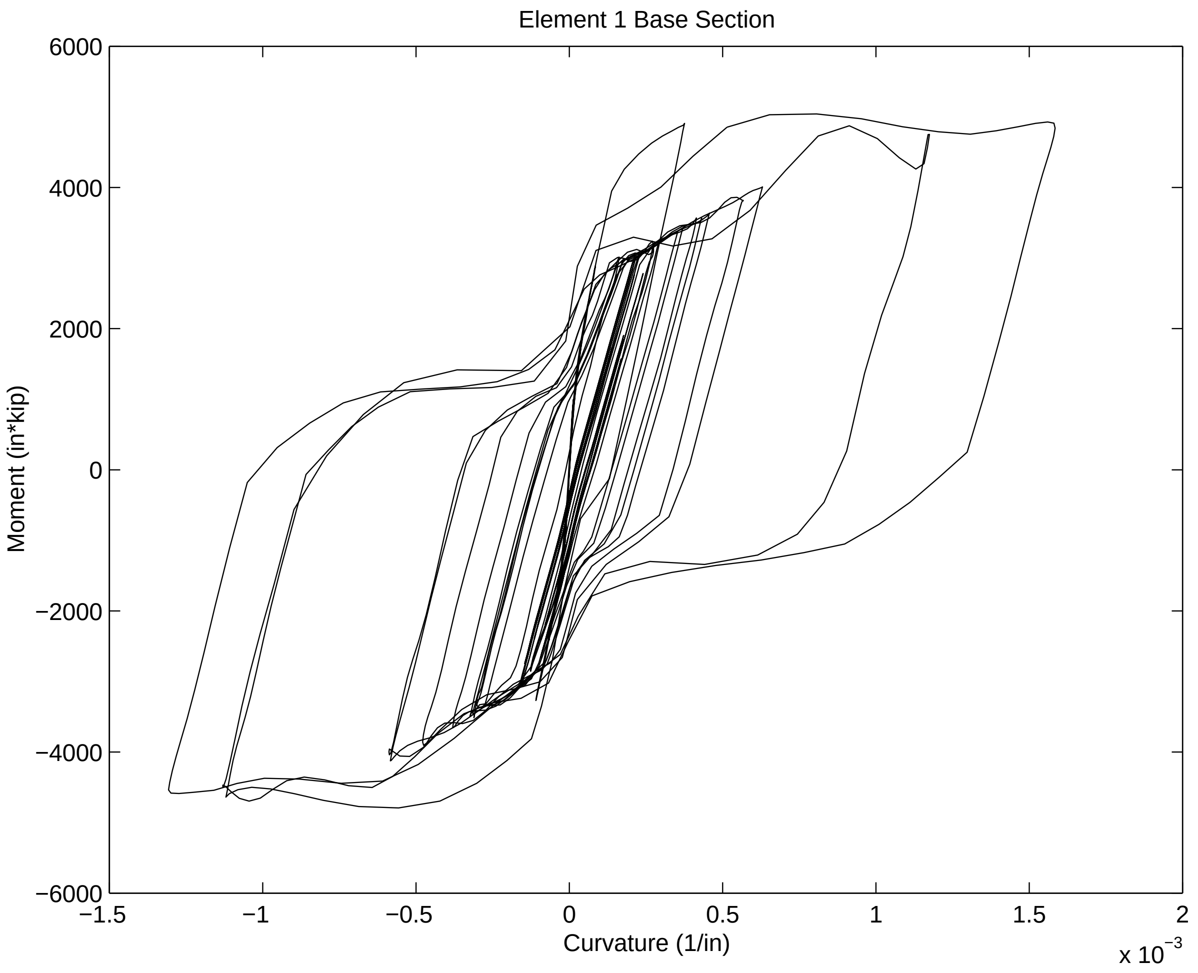

Figure 9: Column section moment-curvature results

4 EXAMPLE 4 - Multibay Two Story Frame Example

In this next example the use of variable substitution and the Tcl loop control structure for building models is demonstrated.

4.1 Example 4.1

This example is of a reinforced concrete multibay two story frame, as shown in figure 10, subject to gravity loads.

Files Required

1. Example4.1.tcl

Model

A model of the frame shown in figure 10 is created. The number of objects in the model is dependent on the parameter numBay. The (numBay +1) $*$ 3) nodes are created, one column line at a time, with the node at the base of the columns fixed in all directions. Three materials are constructed, one for the concrete core, one for the concrete cover and one for the reinforcement steel. Three fiber discretized sections are then built, one for the exterior columns, one for the interior columns and one for the girders. Each of the members in the frame is modelled using nonlinear beam-column elements with 4 (nP) integration points and a linear geometric transformation object.

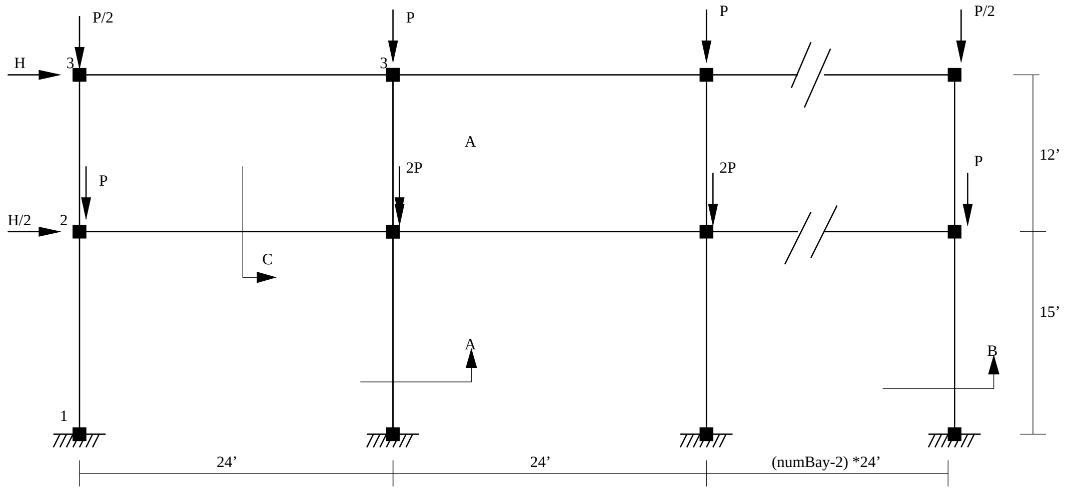

Figure 10: Example 4.1

For gravity loads, a single load pattern with a linear time series and two vertical nodal loads acting at the first and second floor nodes of each column line is used. The load at the lower level is twice that of the upper level and the load on the interior columns is twice that of the exterior columns.

For the lateral load analysis, a second load pattern with a linear time series is introduced after the gravity load analysis. Associated with this load pattern are two nodal loads acting on nodes 2 and 3, with the load level at node 3 twice that acting at node 2.

Analysis

A solution Algorithm of type Newton is created. The solution algorithm uses a ConvergenceTest based on the norm of the displacement increment vector. The integrator for the analysis will be LoadControl with a load step increment of 0.1. The storage for the system of equations is BandGeneral. The equations are numbered using an RCM (reverse Cuthill-McKee) numberer. The constraints are enforced with a Plain constraint handler. Once the components of the analysis have been defined, the analysis object is then created. For this problem a Static analysis object is used and 10 steps are performed to load the model with the desired gravity load.

After the gravity load analysis has been performed, the gravity loads are set to constant and the time in the domain is reset to 0.0. A new LoadControl integrator is now added. The new LoadControl integrator has an initial load step of 1.0, but this can vary between 0.02 and 2.0 depending on the number of iterations required to achieve convergence at each load step. 100 steps are then performed.

Output Specification

For the pushover analysis the lateral displacements at nodes 2 and 3 will be stored in the file Node41.out for post-processing. In addition, if the variable displayMode is set to "displayON" the load-displacement curve for horizontal displacements at node 3 will be displayed in a window on the user's terminal.

OpenSees Script

```
# OpenSees Example 4.1
# OpenSees Primer
#
# Units: kips, in, sec

# Parameter identifying the number of bays
set numBay          3

# ------------------------------
# Start of model generation
# ------------------------------

# Create ModelBuilder (with two-dimensions and 3 DOF/node)
model BasicBuilder -ndm 2 -ndf 3

# Create nodes
# ------------

# Set parameters for overall model geometry
```

```
set bayWidth      288
set nodeID          1

# Define nodes
for {set i 0} {$i <= $numBay} {incr i 1} {
    set xDim [expr $i * $bayWidth]

    #             tag             X   Y
    node           $nodeID    $xDim  0
    node  [expr $nodeID+1]    $xDim 180
    node  [expr $nodeID+2]    $xDim 324

    incr nodeID 3
}

# Fix supports at base of columns
for {set i 0} {$i <= $numBay} {incr i 1} {
#                node  DX   DY   RZ
    fix [expr $i*3+1]   1    1    1
}

# Define materials for nonlinear columns
# ------------------------------------------

# CONCRETE
# Cover concrete
#                  tag -f'c  -epsco  -f'cu -epscu
uniaxialMaterial Concrete01 1 -4.00  -0.002    0.0 -0.006

# Core concrete
uniaxialMaterial Concrete01 2 -5.20  -0.005  -4.70  -0.02

# STEEL
# Reinforcing steel
#                       tag fy   E0     b
uniaxialMaterial Steel01  3  60 30000 0.02


# Define cross-section for nonlinear columns
# ------------------------------------------

# Interior column section - Section A
section Fiber 1 {
   #           mat nfIJ nfJK   yI   zI     yJ   zJ    yK  zK    yL  zL
   patch quadr  2    1   12 -11.5  10 -11.5 -10  11.5 -10  11.5  10
   patch quadr  1    1   14 -13.5 -10 -13.5 -12  13.5 -12  13.5 -10
   patch quadr  1    1   14 -13.5  12 -13.5  10  13.5  10  13.5  12
   patch quadr  1    1    2 -13.5  10 -13.5 -10 -11.5 -10 -11.5  10
   patch quadr  1    1    2  11.5  10  11.5 -10  13.5 -10  13.5  10

   #              mat nBars area    yI zI    yF zF
   layer straight  3    6   1.56 -10.5  9 -10.5 -9
   layer straight  3    6   1.56  10.5  9  10.5 -9
}

# Exterior column section - Section B
section Fiber 2 {
   patch quadr 2 1 10 -10  10 -10 -10  10 -10  10  10
   patch quadr 1 1 12 -12 -10 -12 -12  12 -12  12 -10
   patch quadr 1 1 12 -12  12 -12  10  12  10  12  12
   patch quadr 1 1  2 -12  10 -12 -10 -10 -10 -10  10
   patch quadr 1 1  2  10  10  10 -10  12 -10  12  10
   layer straight 3 6 0.79 -9 9 -9 -9
   layer straight 3 6 0.79  9 9  9 -9
}

# Girder section - Section C
section Fiber 3 {
   patch quadr 1 1 12 -12 9 -12 -9 12 -9 12 9
   layer straight 3 4 1.00 -9 9 -9 -9
   layer straight 3 4 1.00  9 9  9 -9
}


# Define column elements
# ----------------------

# Number of integration points
set nP 4

# Geometric transformation
geomTransf Linear 1

set beamID         1

# Define elements
for {set i 0} {$i <= $numBay} {incr i 1} {
   # set some parameters
   set iNode [expr $i*3 + 1]
   set jNode [expr $i*3 + 2]

   for {set j 1} {$j < 3} {incr j 1} {
      # add the column element (secId == 2 if external, 1 if internal column)
      if {$i == 0} {
         element nonlinearBeamColumn  $beamID   $iNode $jNode  $nP   2    1
      } elseif {$i == $numBay} {
         element nonlinearBeamColumn  $beamID   $iNode $jNode  $nP   2    1
      } else {
         element nonlinearBeamColumn  $beamID   $iNode $jNode  $nP   1    1
      }

      # increment the parameters
      incr iNode  1
      incr jNode  1
      incr beamID 1
   }
}


# Define beam elements
# ----------------------

# Number of integration points
set nP 4

# Geometric transformation
geomTransf Linear 2

# Define elements
for {set j 1} {$j < 3} {incr j 1} {
   # set some parameters
   set iNode [expr $j + 1]
   set jNode [expr $iNode + 3]

   for {set i 1} {$i <= $numBay} {incr i 1} {
      element nonlinearBeamColumn  $beamID   $iNode $jNode    $nP   3      2

      # increment the parameters
      incr iNode  3
      incr jNode  3
      incr beamID 1
   }
}

# Define gravity loads
# --------------------

# Constant gravity load
set P -192

# Create a Plain load pattern with a Linear TimeSeries
```

```
pattern Plain 1 Linear {
   # Create nodal loads at nodes
   for {set i 0} {$i <= $numBay} {incr i 1} {

      # set some parameters
      set node1 [expr $i*3 + 2]
      set node2 [expr $node1 + 1]

      if {$i == 0} {
         load $node1 0.0            $P  0.0
         load $node2 0.0 [expr $P/2.0]  0.0
      } elseif {$i == $numBay} {
         load $node1 0.0            $P  0.0
         load $node2 0.0 [expr $P/2.0]  0.0
      } else {
         load $node1 0.0 [expr 2.0*$P]  0.0
         load $node2 0.0            $P  0.0
      }
   }
}

# ------------------------------
# End of model generation
# ------------------------------


# ------------------------------------------------
# Start of analysis generation for gravity analysis
# -------------------------------------------------

# Create the convergence test, the norm of the residual with a tolerance of
# 1e-12 and a max number of iterations of 10
test NormDispIncr 1.0e-8  10 0

# Create the solution algorithm, a Newton-Raphson algorithm
algorithm Newton

# Create the integration scheme, the LoadControl scheme using steps of 0.1
integrator LoadControl 0.1 1 0.1 0.1

# Create the system of equation, a SPD using a profile storage scheme
system BandGeneral

# Create the DOF numberer, the reverse Cuthill-McKee algorithm
numberer RCM

# Create the constraint handler, the transformation method
```

```
constraints Plain

# Create the analysis object
analysis Static


# ----------------------------------------------
# End of analysis generation for gravity analysis
# -----------------------------------------------


# ------------------------------
# Perform gravity load analysis
# ------------------------------

# perform the gravity load analysis, requires 10 steps to reach the load level
analyze 10


# set gravity loads to be const and set pseudo time to be 0.0
#  for start of lateral load analysis
loadConst -time 0.0


# ------------------------------
# Add lateral loads
# ------------------------------

# Reference lateral load for pushover analysis
set H   10

# Reference lateral loads
# Create a Plain load pattern with a Linear TimeSeries
pattern Plain 2 Linear {
    load 2 [expr $H/2.0]  0.0  0.0
    load 3            $H  0.0  0.0
}

# ------------------------------
# Start of recorder generation
# ------------------------------

# Create a recorder which writes to Node.out and prints
# the current load factor (pseudo-time) and dof 1 displacements at node 2 & 3
recorder Node Node41.out disp -time -node 2 3 -dof 1

# Source in some commands to display the model
# comment out one of lines
set displayMode "displayON"
#set displayMode "displayOFF"

if {$displayMode == "displayON"} {
    # a window to plot the nodal displacements versus load for node 3
    recorder plot Node41.out Node3Xdisp 10 340 300 300 -columns 3 1
}

# ------------------------------
# End of recorder generation
# ------------------------------


# ------------------------------
# Start of lateral load analysis
# ------------------------------

# Change the integrator to take a min and max load increment
integrator LoadControl 1.0 4 0.02 2.0

# Perform the analysis

# Perform the pushover analysis
# Set some parameters
set maxU 15.0;         # Max displacement
set controlDisp 0.0;
set ok 0;

while {$controlDisp < $maxU && $ok == 0} {
    set ok [analyze 1]
    set controlDisp [nodeDisp 3 1]

    if {$ok != 0} {
puts "... trying an initial tangent iteration with Newton"
test NormDispIncr 1.0e-8  4000 0
  algorithm Newton -initial
set ok [analyze 1]
test NormDispIncr 1.0e-8  10 0
algorithm Newton
    }

    if {$ok != 0} {
puts "... that failed .. trying Broyden"
test NormDispIncr 1.0e-8  10 1
  algorithm Broyden
set ok [analyze 1]
test NormDispIncr 1.0e-8  10 0
algorithm Newton
    }

    if {$ok != 0} {
puts "... that failed .. trying an initial tangent with Broyden"
test NormDispIncr 1.0e-8  10 1
  algorithm Broyden 20 -initial
set ok [analyze 1]
test NormDispIncr 1.0e-8  10 0
algorithm Newton
    }
}

if {$ok != 0} {
    puts "Pushover analysis FAILED"
} else {
    puts "Pushover analysis completed SUCCESSFULLY"
}
```

Results

The output consists of the file Node41.out containing a line for each step of the lateral load analysis. Each line contains the load factor, the lateral displacements at nodes 2 and 3. A plot of the load-displacement curve for the frame is given in figure 11.

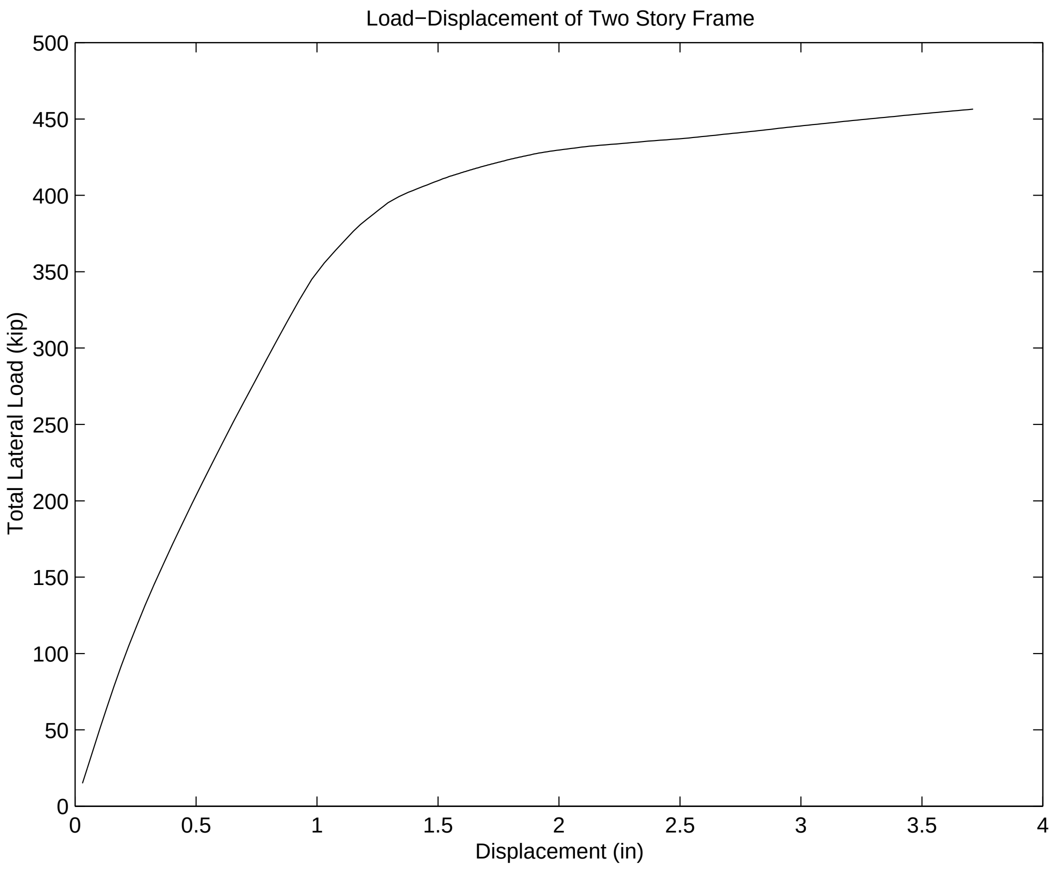

Figure 11: Pushover curve for two-story three-bay frame

5 EXAMPLE 5 - Three-Dimensional Rigid Frame

5.1 Example 5.1

This example is of a three-dimensional reinforced concrete rigid frame, as shown in figure 12, subjected to bi-directional earthquake ground motion.

Files Required

1. Example5.1.tcl
2. RCsection.tcl
3. tabasFN.txt
4. tabasFP.txt

Model

A model of the rigid frame shown in figure 12 is created. The model consists of three stories and one bay in each direction. Rigid diaphragm multi-point constraints are used to enforce the rigid in-plane stiffness assumption for the floors. Gravity loads are applied to the structure and the 1978 Tabas acceleration records are the uniform earthquake excitations.

Nonlinear beam column elements are used for all members in the structure. The beam sections are elastic while the column sections are discretized by fibers of concrete and steel. Elastic beam column elements may have been used for the beam members; but, it is useful to see that section models other than fiber sections may be used in the nonlinear beam column element.

Analysis

A solution Algorithm of type Newton is used for the nonlinear problem. The solution algorithm uses a ConvergenceTest which tests convergence on the norm of the energy increment vector. The integrator for this analysis will be of type Newmark with a γ of 0.25 and a β of 0.5. Due to the presence of the multi-point constraints, a Transformation constraint handler is used. The equations are formed using a sparse storage scheme which will perform pivoting during the equation solving, so the System is SparseGeneral. As SparseGeneral will perform it's own internal numbering of the equations, a Plain numberer is used which simply assigns equation numbers to the degrees-of-freedom.

Once all the components of an analysis are defined, the Analysis object itself is created. For this problem a Transient Analysis object is used. 2000 steps are performed with a time step of 0.01.

Output Specification

The nodal displacements at nodes 9, 14, and 19 (the master nodes for the rigid diaphragms) will be stored in the file node51.out for post-processing.

OpenSees Script

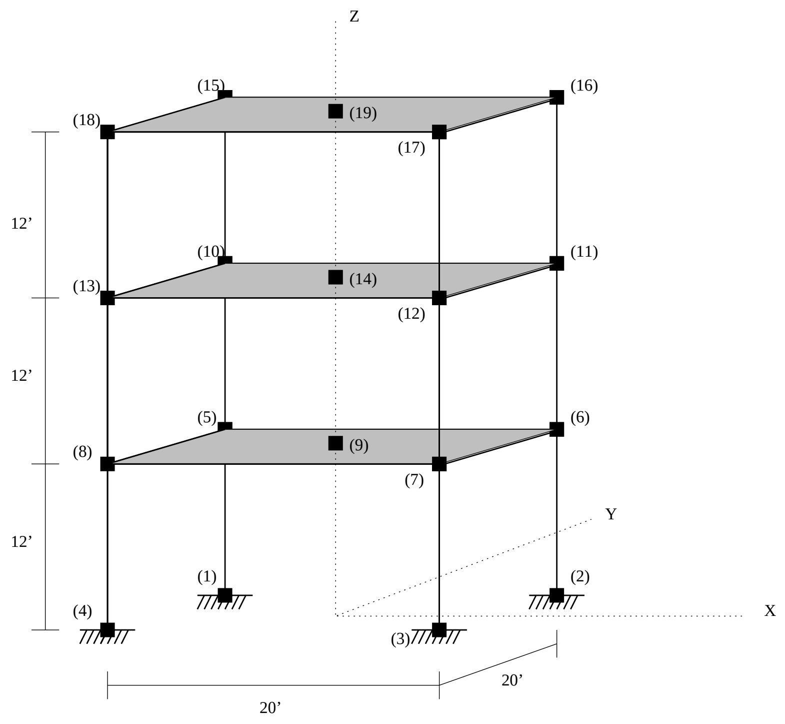

Figure 12: Example 5.1

```
# OpenSees Example 5.1
# OpenSees Primer
#
# Units: kips, in, sec

# ----------------------------
# Start of model generation
# ----------------------------

# Create ModelBuilder with 3 dimensions and 6 DOF/node
model BasicBuilder -ndm 3 -ndf 6

# Define geometry
# ---------------

# Set parameters for model geometry
set h 144.0;       # Story height
set by 240.0;      # Bay width in Y-direction
set bx 240.0;      # Bay width in X-direction

# Create nodes
#    tag             X             Y           Z
node  1  [expr -$bx/2] [expr  $by/2]           0
node  2  [expr  $bx/2] [expr  $by/2]           0
node  3  [expr  $bx/2] [expr -$by/2]           0
node  4  [expr -$bx/2] [expr -$by/2]           0

node  5  [expr -$bx/2] [expr  $by/2]          $h
node  6  [expr  $bx/2] [expr  $by/2]          $h
node  7  [expr  $bx/2] [expr -$by/2]          $h
node  8  [expr -$bx/2] [expr -$by/2]          $h

node 10  [expr -$bx/2] [expr  $by/2] [expr 2*$h]
node 11  [expr  $bx/2] [expr  $by/2] [expr 2*$h]
node 12  [expr  $bx/2] [expr -$by/2] [expr 2*$h]
node 13  [expr -$bx/2] [expr -$by/2] [expr 2*$h]

node 15  [expr -$bx/2] [expr  $by/2] [expr 3*$h]
node 16  [expr  $bx/2] [expr  $by/2] [expr 3*$h]
node 17  [expr  $bx/2] [expr -$by/2] [expr 3*$h]
node 18  [expr -$bx/2] [expr -$by/2] [expr 3*$h]

# Master nodes for rigid diaphragm
#    tag X Y          Z
node  9  0 0         $h
node 14  0 0 [expr 2*$h]
node 19  0 0 [expr 3*$h]

# Set base constraints
#   tag DX DY DZ RX RY RZ
fix  1   1  1  1  1  1  1
fix  2   1  1  1  1  1  1
fix  3   1  1  1  1  1  1
fix  4   1  1  1  1  1  1

# Define rigid diaphragm multi-point constraints
#               normalDir  master     slaves
rigidDiaphragm     3          9     5  6  7  8
rigidDiaphragm     3         14    10 11 12 13
rigidDiaphragm     3         19    15 16 17 18

# Constraints for rigid diaphragm master nodes
#   tag DX DY DZ RX RY RZ
fix  9   0  0  1  1  1  0
fix 14   0  0  1  1  1  0
fix 19   0  0  1  1  1  0

# Define materials for nonlinear columns
# ------------------------------------------
# CONCRETE
# Core concrete (confined)
#                           tag  f'c  epsc0  f'cu  epscu
uniaxialMaterial Concrete01  1  -5.0 -0.005  -3.5  -0.02

# Cover concrete (unconfined)
set fc 4.0
uniaxialMaterial Concrete01  2   -$fc -0.002   0.0 -0.006

# STEEL
# Reinforcing steel
#                        tag fy   E     b
uniaxialMaterial Steel01  3  60 30000 0.02

# Column width
set d 18.0

# Source in a procedure for generating an RC fiber section
source RCsection.tcl

# Call the procedure to generate the column section
#         id  h  b cover core cover steel nBars area nfCoreY nfCoreZ nfCoverY nfCoverZ
RCsection  1 $d $d   2.5    1     2     3     3 0.79       8       8       10       10

# Concrete elastic stiffness
set E [expr 57000.0*sqrt($fc*1000)/1000];

# Column torsional stiffness
set GJ 1.0e10;

# Linear elastic torsion for the column
uniaxialMaterial Elastic 10 $GJ

# Attach torsion to the RC column section
#                 tag uniTag uniCode       secTag
section Aggregator 2    10      T    -section 1
set colSec 2
# Define column elements
# ----------------------

#set PDelta "ON"
set PDelta "OFF"

# Geometric transformation for columns
if {$PDelta == "ON"} {
   #                            tag  vecxz
   geomTransf LinearWithPDelta  1   1 0 0
} else {
   geomTransf Linear  1   1 0 0
}

# Number of column integration points (sections)
set np 4

# Create the nonlinear column elements
#                           tag ndI ndJ nPts   secID  transf
element nonlinearBeamColumn  1   1   5   $np  $colSec    1
element nonlinearBeamColumn  2   2   6   $np  $colSec    1
element nonlinearBeamColumn  3   3   7   $np  $colSec    1
element nonlinearBeamColumn  4   4   8   $np  $colSec    1

element nonlinearBeamColumn  5   5  10   $np  $colSec    1
element nonlinearBeamColumn  6   6  11   $np  $colSec    1
element nonlinearBeamColumn  7   7  12   $np  $colSec    1
element nonlinearBeamColumn  8   8  13   $np  $colSec    1

element nonlinearBeamColumn  9  10  15   $np  $colSec    1
element nonlinearBeamColumn 10  11  16   $np  $colSec    1
element nonlinearBeamColumn 11  12  17   $np  $colSec    1
element nonlinearBeamColumn 12  13  18   $np  $colSec    1

# Define beam elements
# --------------------

# Define material properties for elastic beams
# Using beam depth of 24 and width of 18
# --------------------------------------------
set Abeam [expr 18*24];
# "Cracked" second moments of area
set Ibeamzz [expr 0.5*1.0/12*18*pow(24,3)];
set Ibeamyy [expr 0.5*1.0/12*24*pow(18,3)];

# Define elastic section for beams
#               tag  E    A      Iz       Iy     G   J
section Elastic  3  $E $Abeam $Ibeamzz $Ibeamyy $GJ 1.0
set beamSec 3

# Geometric transformation for beams
#                tag  vecxz
geomTransf Linear 2   1 1 0

# Number of beam integration points (sections)
set np 3

# Create the beam elements
#                           tag ndI  ndJ nPts    secID   transf
element nonlinearBeamColumn  13  5   6   $np  $beamSec     2
element nonlinearBeamColumn  14  6   7   $np  $beamSec     2
element nonlinearBeamColumn  15  7   8   $np  $beamSec     2
element nonlinearBeamColumn  16  8   5   $np  $beamSec     2

element nonlinearBeamColumn  17 10  11   $np  $beamSec     2
element nonlinearBeamColumn  18 11  12   $np  $beamSec     2
element nonlinearBeamColumn  19 12  13   $np  $beamSec     2
element nonlinearBeamColumn  20 13  10   $np  $beamSec     2

element nonlinearBeamColumn  21 15  16   $np  $beamSec     2
element nonlinearBeamColumn  22 16  17   $np  $beamSec     2
element nonlinearBeamColumn  23 17  18   $np  $beamSec     2
element nonlinearBeamColumn  24 18  15   $np  $beamSec     2

# Define gravity loads
# --------------------
# Gravity load applied at each corner node
# 10% of column capacity
set p [expr 0.1*$fc*$h*$h]

# Mass lumped at master nodes
set g 386.4;            # Gravitational constant
set m [expr (4*$p)/$g]

# Rotary inertia of floor about master node
set i [expr $m*($bx*$bx+$by*$by)/12.0]

# Set mass at the master nodes
#    tag MX MY MZ RX RY RZ
mass  9  $m $m  0  0  0 $i
mass 14  $m $m  0  0  0 $i
mass 19  $m $m  0  0  0 $i

# Define gravity loads
pattern Plain 1 Constant {
   foreach node {5 6 7 8  10 11 12 13  15 16 17 18} {
      load $node 0.0 0.0 -$p 0.0 0.0 0.0
   }
}

# Define earthquake excitation
# ----------------------------
# Set up the acceleration records for Tabas fault normal and fault parallel
set tabasFN "Path -filePath tabasFN.txt -dt 0.02 -factor $g"
set tabasFP "Path -filePath tabasFP.txt -dt 0.02 -factor $g"
```

```
# Define the excitation using the Tabas ground motion records
#                         tag dir         accel series args
pattern UniformExcitation  2   1  -accel      $tabasFN
pattern UniformExcitation  3   2  -accel      $tabasFP


# -----------------------
# End of model generation
# -----------------------



# ----------------------------
# Start of analysis generation
# ----------------------------

# Create the convergence test
#                tol   maxIter  printFlag
test EnergyIncr 1.0e-8   20         3

# Create the solution algorithm
algorithm Newton

# Create the system of equation storage and solver
system SparseGeneral -piv

# Create the constraint handler
constraints Transformation

# Create the time integration scheme
#                   gamma beta
integrator Newmark   0.5  0.25

# Create the DOF numberer
numberer RCM

# Create the transient analysis
analysis Transient

# --------------------------
# End of analysis generation
# --------------------------



# ----------------------------
# Start of recorder generation
# ----------------------------
# Record DOF 1 and 2 displacements at nodes 9, 14, and 19
recorder Node node51.out disp -time -node 9 14 19 -dof 1 2

# --------------------------
# End of recorder generation
# --------------------------



# --------------------
# Perform the analysis
# --------------------

# Analysis duration of 20 seconds
#       numSteps  dt
analyze   2000   0.01
```

Results

The results consist of the file node.out, which contains a line for every time step. Each line contains the time and the horizontal and vertical displacements at the diaphragm master nodes (9, 14 and 19) i.e. time Dx9 Dy9 Dx14 Dy14 Dx19 Dy19. The horizontal displacement time history of the first floor diaphragm node 9 is shown in figure 13. Notice the increase in period after about 10 seconds of earthquake excitation, when the large pulse in the ground motion propogates through the structure. The displacement profile over the three stories shows a soft-story mechanism has formed in the first floor columns. The numerical solution converges even though the drift is $\approx 20\%$. The inclusion of P-Delta effects shows structural collapse under such large drifts.

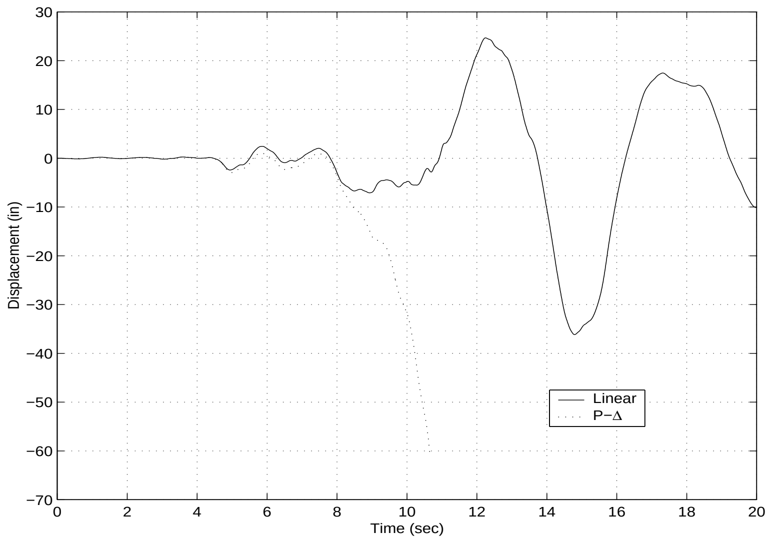

Figure 13: Node 9 displacement time history

6 EXAMPLE 6 - Simply Supported Beam

In this example a simple problem in solid dynamics is considered. The structure is a simply supported beam modelled with two dimensional solid elements.

Files Required

1. Example6.1.tcl

Model

For two dimensional analysis, a typical solid element is defined as a volume in two dimensional space. Each node of the analysis has two displacement degrees of freedom. Thus the model is defined with $ndm := 2$ and $ndf := 2$. pp For this model, a mesh is generated using the "block2D" command. The number of nodes in the local x-direction of the block is nx and the number of nodes in the local y-direction of the block is ny. The block2D generation nodes {1,2,3,4} are prescribed to define the two dimensional domain of the beam, which is of size 40×10.

Three possible quadrilateral elements can be used for the analysis. These may be created using the terms "bbarQuad," "enhancedQuad" or "quad." This is a plane strain problem. An elastic isotropic material is used.

For initial gravity load analysis, a single load pattern with a linear time series and two vertical nodal loads are used.

Analysis

A solution algorithm of type Newton is used for the problem. The solution algorithm uses a ConvergenceTest which tests convergence on the norm of the energy increment vector. Ten static load steps are performed.

Subsequent to the static analysis, the wipeAnalysis and remove loadPatern commands are used to remove the nodal loads and create a new analysis. The nodal displacements have not changed. However, with the external loads removed the structure is no longer in static equilibrium.

The integrator for the dynamic analysis if of type GeneralizedMidpoint with $\alpha := 0.5$. This choice is uconditionally stable and energy conserving for linear problems. Additionally, this integrator conserves linear and angular momentum for both linear and non-linear problems. The dynamic analysis is performed using 100 time increments with a time step $\Delta t := 0.50$.

OpenSees Script

```
# ----------------------------
# Start of model generation
# ----------------------------

# Create ModelBuilder with 3 dimensions and 6 DOF/node
model basic -ndm 2 -ndf 2
```

```
# create the material
nDMaterial ElasticIsotropic   1   1000   0.25  6.75


# Define geometry
# ---------------

# define some  parameters

set Quad  quad
set Quad  bbarQuad
set Quad  enhancedQuad

if {$Quad == "enhancedQuad" } {
    set eleArgs "PlaneStrain2D  1"
}

if {$Quad == "quad" } {
    set eleArgs "1 PlaneStrain2D  1"
}

if {$Quad == "bbarQuad" } {
    set eleArgs "1"
}

set nx 8; # NOTE: nx MUST BE EVEN FOR THIS EXAMPLE
set ny 2
set bn [expr $nx + 1 ]
set l1 [expr $nx/2 + 1 ]
set l2 [expr $l1 + $ny*($nx+1) ]

# now create the nodes and elements using the block2D command
block2D $nx $ny   1 1  $Quad  $eleArgs {
    1   0   0
    2  40   0
    3  40  10
    4   0  10
}

# Single point constraints
#   node   u1  u2
fix    1    1    1
fix  $bn    0    1

# Gravity loads
pattern Plain 1 Linear {
    load $l1  0.0  -1.0
    load $l2  0.0  -1.0
}

# --------------------------------------------------------------------
# Start of static analysis (creation of the analysis & analysis itself)
# --------------------------------------------------------------------

# Load control with variable load steps
#                       init   Jd  min   max
integrator LoadControl  1.0  1   1.0   10.0

# Convergence test
#                  tolerance maxIter displayCode
test EnergyIncr  1.0e-12    10         0

# Solution algorithm
algorithm Newton

# DOF numberer
numberer RCM

# Cosntraint handler
constraints Plain

# System of equations solver
system ProfileSPD

# Analysis for gravity load
analysis Static

# Perform the analysis
analyze   10


# -------------------------
# End of static analysis
# -------------------------

# ---------------------------
# Start of recorder generation
# ---------------------------


recorder Node Node.out  disp -time -node $l1 -dof 2
recorder plot Node.out CenterNodeDisp 625 10 625 450 -columns 1 2
```

```
# create the display
recorder display g3 10 10 800 200 -wipe
prp 20 5.0 100.0
vup 0 1 0
viewWindow -30 30 -10 10
display 10 0 5

# --------------------------
# End of recorder generation
# --------------------------



# ---------------------------------------
# Create and Perform the dynamic analysis
# ---------------------------------------

# Remove the static analysis & reset the time to 0.0
wipeAnalysis
setTime 0.0

# Now remove the loads and let the beam vibrate
remove loadPattern 1

# Create the transient analysis
test EnergyIncr  1.0e-12    10         0
algorithm Newton
numberer RCM
constraints Plain
integrator Newmark 0.5 0.25
#integrator GeneralizedMidpoint 0.50
analysis Transient

# Perform the transient analysis (50 sec)
#       numSteps  dt
analyze  100     0.5
```

Results

The results consist of the file Node.out, which contains a line for every time step. Each line contains the time and the vertical displacement at the bottom center of the beam. The time history is shown in figure 14.

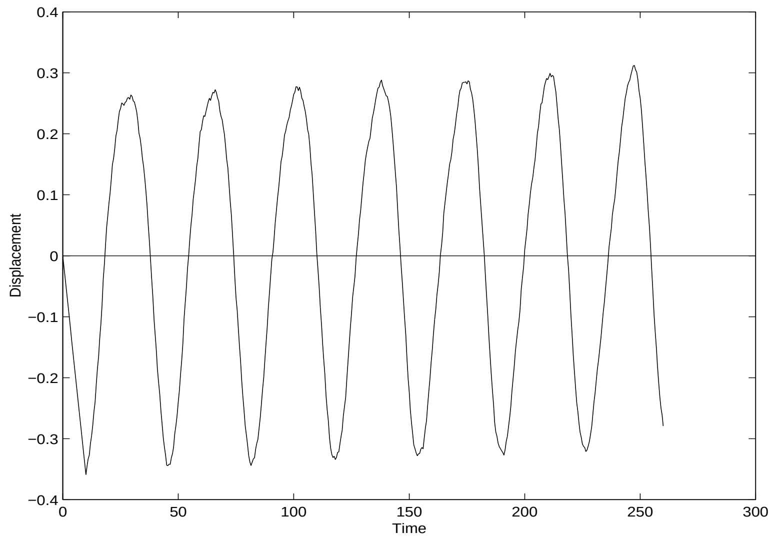

Figure 14: Displacement vs. Time for Bottom Center of Beam

7 EXAMPLE 7 - Dynamic Shell Analysis

In this example a simple problem in shell dynamics is considered. The structure is a curved hoop shell structure that looks like the roof of a Safeway.

Files Required

1. Example7.1.tcl

Model

For shell analysis, a typical shell element is defined as a surface in three dimensional space. Each node of a shell analysis has six degrees of freedom, three displacements and three rotations. Thus the model is defined with $ndm := 3$ and $ndf := 6$.

For this model, a mesh is generated using the "block2D" command. The number of nodes in the local x-direction of the block is nx and the number of nodes in the local y-direction of the block is ny. The block2D generation nodes {1,2,3,4, 5,7,9} are defined such that the structure is curved in three dimensional space.

The OpenSees shell element is constructed using the command "ShellMITC4". An elastic membrane-plate material section model, appropriate for shell analysis, is constructed using the "ElasticMembranePlateSection" command. In this case, the elastic modulus $E := 3.0e3$, Poisson's ratio $\nu := 0.25$, the thickness $h := 1.175$ and the mass density per unit volume $\rho := 1.27$

For initial gravity load analysis, a single load pattern with a linear time series and three vertical nodal loads are used.

Boundary conditions are applied using the fixZ command. In this case, all the nodes whose z-coordiate is 0.0 have the boundary condition {1,1,1, 0,1,1}. All degrees-of-freedom are fixed except rotation about the x-axis, which is free. The same boundary conditions are applied where the z-coordinate is 40.0.

Analysis

A solution algorithm of type Newton is used for the problem. The solution algorithm uses a ConvergenceTest which tests convergence on the norm of the energy increment vector. Five static load steps are performed.

Subsequent to the static analysis, the wipeAnalysis and remove loadPatern commands are used to remove the nodal loads and create a new analysis. The nodal displacements have not changed. However, with the external loads removed the structure is no longer in static equilibrium.

The integrator for the dynamic analysis if of type GeneralizedMidpoint with $\alpha := 0.5$. This choice is uconditionally stable and energy conserving for linear problems. Additionally, this integrator conserves linear and angular momentum for both linear and non-linear problems. The dynamic analysis is performed using 250 time increments with a time step $\Delta t := 0.50$.

OpenSees Script

```
# ----------------------------
# Start of model generation
# ----------------------------
model basic -ndm 3 -ndf 6

# create the material
section ElasticMembranePlateSection  1   3.0e3  0.25  1.175  1.27

# set some parameters for node and element generation
set Plate ShellMITC4

set eleArgs "1"

#these should both be even
set nx 8
set ny 2

#loaded nodes
set mid [expr (  ($nx+1)*($ny+1)+1 ) / 2 ]
set side1 [expr ($nx + 2)/2 ]
set side2 [expr ($nx+1)*($ny+1) - $side1 + 1 ]

# generate the nodes and elements
block2D $nx $ny 1 1 $Plate $eleArgs {
    1   -20    0     0
    2   -20    0    40
    3    20    0    40
    4    20    0     0
    5   -10   10    20
    7    10   10    20
    9    0    10    20
}

# add some loads
pattern Plain 1 Linear {
    load $mid    0.0  -0.5   0.0   0.0   0.0  0.0
    load $side1  0.0  -0.25  0.0   0.0   0.0  0.0
    load $side2  0.0  -0.25  0.0   0.0   0.0  0.0
}


# define the boundary conditions
# rotation free about x-axis (remember right-hand-rule)
fixZ 0.0   1 1 1  0 1 1
fixZ 40.0  1 1 1  0 1 1

# Load control with variable load steps
#                       init   Jd  min   max
integrator LoadControl  1.0  1   1.0   10.0

# Convergence test
#                  tolerance maxIter displayCode
test EnergyIncr     1.0e-10    20         1

# Solution algorithm
algorithm Newton

# DOF numberer
numberer RCM

# Cosntraint handler
constraints Plain

# System of equations solver
system SparseGeneral -piv
#system ProfileSPD

# Analysis for gravity load
#analysis Transient
analysis Static


# Perform the gravity load analysis
analyze 5


# --------------------------
# End of static analysis
# --------------------------

# ----------------------------
# Start of recorder generation
# ----------------------------

recorder Node Node.out  disp -time -node $mid -dof 2
recorder plot Node.out CenterNodeDisp 625 10 625 450 -columns 1 2
recorder display shellDynamics 10 10 600 600 -wipe
prp -100 20 30
vup 0 1 0
display 1 0 100

# --------------------------
# End of recorder generation
# --------------------------
# ----------------------------------------
# Create and Perform the dynamic analysis
# ----------------------------------------

# Remove the static analysis & reset the time to 0.0
wipeAnalysis
setTime 0.0

# Now remove the loads and let the beam vibrate
remove loadPattern 1

# Create the transient analysis
test EnergyIncr     1.0e-10    20          1
algorithm Newton
numberer RCM
constraints Plain
#integrator GeneralizedMidpoint 0.50
integrator Newmark 0.50 0.25
analysis Transient

# Perform the transient analysis
analyze 250 0.5
```

Results

The results consist of the file Node.out, which contains a line for every time step. Each line contains the time and the vertical displacement at the upper center of the hoop structure. The time history is shown in figure 15.

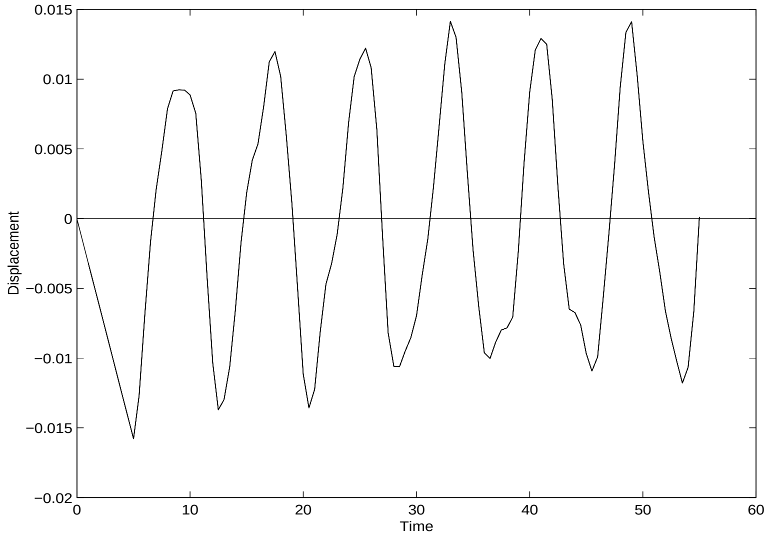

Figure 15: Displacement vs. Time for Top Center of Hoop Structure

8 EXAMPLE 8 - Cantilever Beam

In this example a simple problem in solid dynamics is considered. The structure is a cantilever beam modelled with three dimensional solid elements.

Files Required

1. Example8.1.tcl

Model

For three dimensional analysis, a typical solid element is defined as a volume in three dimensional space. Each node of the analysis has three displacement degrees of freedom. Thus the model is defined with $ndm := 3$ and $ndf := 3$.

For this model, a mesh is generated using the "block3D" command. The number of nodes in the local x-direction of the block is nx, the number of nodes in the local y-direction of the block is ny and the number of nodes in the local z-direction of the block is nz. The block3D generation nodes {1,2,3,4,5,6,7,8} are prescribed to define the three dimensional domain of the beam, which is of size $2 \times 2 \times 10$.

Two possible brick elements can be used for the analysis. These may be created using the terms "stdBrick" or "bbarBrick." An elastic isotropic material is used.

For initial gravity load analysis, a single load pattern with a linear time series and a single nodal loads is used.

Boundary conditions are applied using the fixZ command. In this case, all the nodes whose z-coordiate is 0.0 have the boundary condition {1,1,1}, fully fixed.

Analysis

A solution algorithm of type Newton is used for the problem. The solution algorithm uses a ConvergenceTest which tests convergence on the norm of the energy increment vector. Five static load steps are performed.

Subsequent to the static analysis, the wipeAnalysis and remove loadPatern commands are used to remove the nodal loads and create a new analysis. The nodal displacements have not changed. However, with the external loads removed the structure is no longer in static equilibrium.

The integrator for the dynamic analysis if of type GeneralizedMidpoint with $\alpha := 0.5$. This choice is uconditionally stable and energy conserving for linear problems. Additionally, this integrator conserves linear and angular momentum for both linear and non-linear problems. The dynamic analysis is performed using 100 time increments with a time step $\Delta t := 2.0$.

OpenSees Script

```
# ---------------------------
# Start of model generation
# ----------------------------

# Create ModelBuilder with 3 dimensions and 6 DOF/node
model basic -ndm 3 -ndf 3


# create the material
nDMaterial ElasticIsotropic   1   100   0.25  1.27


# Define geometry
# ---------------

# define some  parameters
set eleArgs "1"

set element stdBrick
#set element BbarBrick

set nz 6
set nx 2
set ny 2

set nn [expr ($nz+1)*($nx+1)*($ny+1) ]

# mesh generation
block3D $nx $ny $nz   1 1  $element  $eleArgs {
    1   -1     -1      0
    2    1     -1      0
    3    1      1      0
    4   -1      1      0
    5   -1     -1     10
    6    1     -1     10
    7    1      1     10
    8   -1      1     10
}


set load 0.10

# Constant point load
pattern Plain 1 Linear {
   load $nn  $load  $load  0.0
}

# boundary conditions
fixZ 0.0   1 1 1
```

```
# ----------------------------------------------------------------------
# Start of static analysis (creation of the analysis & analysis itself)
# ----------------------------------------------------------------------

# Load control with variable load steps
#                       init   Jd  min   max
integrator LoadControl  1.0  1   1.0   10.0

# Convergence test
#                  tolerance maxIter displayCode
test NormUnbalance     1.0e-10    20         1

# Solution algorithm
algorithm Newton

# DOF numberer
numberer RCM

# Cosntraint handler
constraints Plain

# System of equations solver
system ProfileSPD

# Analysis for gravity load
analysis Static

# Perform the analysis
analyze 5

# ------------------------
# End of static analysis
# ------------------------

# --------------------------
# Start of recorder generation
# --------------------------

recorder Node Node.out  disp -time -node $nn -dof 1
recorder plot Node.out CenterNodeDisp 625 10 625 450 -columns 1 2

recorder display ShakingBeam 0 0 300 300 -wipe
prp -100 100 120.5
vup 0 1 0
display 1 0 1
```

```
# --------------------------
# End of recorder generation
# --------------------------


# ---------------------------------------
# Create and Perform the dynamic analysis
# ---------------------------------------

# Remove the static analysis & reset the time to 0.0
wipeAnalysis
setTime 0.0

# Now remove the loads and let the beam vibrate
remove loadPattern 1

# Create the transient analysis
test EnergyIncr     1.0e-10    20         1
algorithm Newton
numberer RCM
constraints Plain
integrator Newmark 0.5 0.25
#integrator GeneralizedMidpoint 0.50
analysis Transient


# Perform the transient analysis (20 sec)
#       numSteps  dt
analyze 100 2.0
}
```

Results

The results consist of the file cantilever.out, which contains a line for every time step. Each line contains the time and the horizontal displacement at the upper right corner the beam. The time history is shown in figure 16.

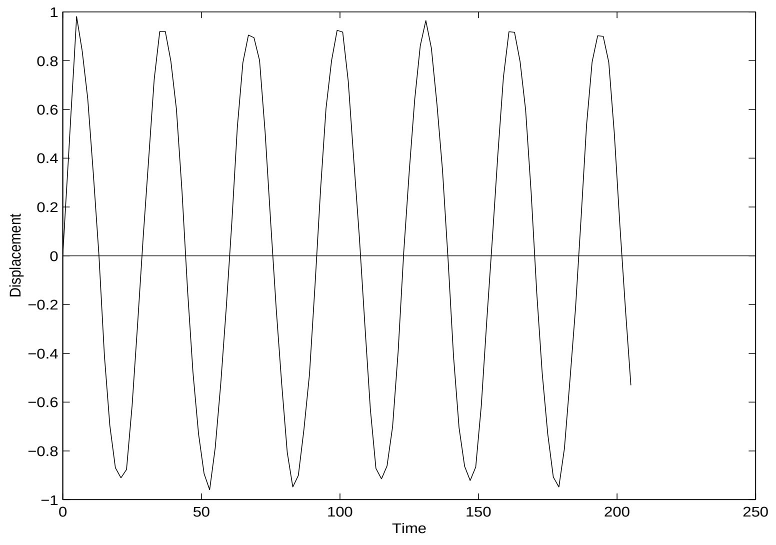

Figure 16: Displacement vs. Time for Upper Right Corner of Beam