Moment Frame Template Accounting for Panel Zone Deformations using OpenSees

Matthew Steward

A project submitted to the faculty of Brigham Young University in partial fulfillment of the requirements for the degree of Master of Science

Paul Richards, Chair
Fernando Fonseca
Richard Balling

Department of Civil & Environmental Engineering
Brigham Young University
June 2015

Copyright © 2015 Matthew Steward
All Rights Reserved
ABSTRACT

Moment Frame Template Accounting for Panel Zone Deformations using OpenSees

Matthew Steward
Department of Civil & Environmental Engineering, BYU
Master of Science

A template has been created for the program OpenSees which can be used to model moment resisting frames. The template accounts for deformations in the panel zone by using the parallelogram model. The code for the template is written in Tool Command Language (TCL), and includes commands specific to OpenSees. Output from computer models were checked against equations for the theoretical models and found to be within 15 percent for story heights above 10 feet.
# TABLE OF CONTENTS

**LIST OF TABLES**........................................................................................................................................... vi

**LIST OF FIGURES**........................................................................................................................................ viii

1  Introduction.................................................................................................................................................. 1

2  Background .................................................................................................................................................. 2

   2.1 Moment Frames ...................................................................................................................................... 2

   2.2 Computer Models ................................................................................................................................... 3

   2.3 OpenSees ............................................................................................................................................... 4

   2.4 The Parallelogram Model ..................................................................................................................... 4

3  Template Creation ....................................................................................................................................... 6

   3.1 Preparation for Defining Elements ........................................................................................................ 6

   3.2 Defining Elements ................................................................................................................................. 7

   3.3 Defining Connections ............................................................................................................................. 11

4  Template Verification .................................................................................................................................. 14

5  Using the input generator ............................................................................................................................ 18

6  Template output .......................................................................................................................................... 19

7  Template Limitations .................................................................................................................................. 20

8  Conclusions .................................................................................................................................................. 21

REFERENCES.................................................................................................................................................... 23

Appendix A.  Parallelogram Template OpenSees Code...................................................................................... 25
LIST OF TABLES

Table 1: Comparison of Building Stiffness from OpenSees Template and Theoretical Model .................................................................16
LIST OF FIGURES

Figure 1: Example of a Moment Frame ................................................................. 3
Figure 2: Example of Moment Connection with Reduced Beam Section (Hamburger et al. 2009) ................................................................. 3
Figure 3: Point and Centerline Model vs. Parallelogram Model ........................... 5
Figure 4: Code Defining Analysis Type for Columns ............................................. 6
Figure 5: Code Defining Horizontal Locations for the Template ........................... 7
Figure 6: Example of Code Defining Nodes and Elements ................................. 10
Figure 7: Sub-function Defining Panel Zone Springs ........................................... 12
Figure 8: Different Contributions to Story Drift (Haburger et al. 2009) ................. 15
Figure 9: Screenshot of Moment Frame Designer with OS Input Button ............... 18
Figure 10: Code Defining Output Information ..................................................... 19
1 INTRODUCTION

The purpose of this project was to create an OpenSees template for moment frames that accounts for deformations in the panel zone. In addition to the main goal of the project, a spreadsheet executable was developed to facilitate inputting data for the OpenSees Template.

This report will describe the need for this template, how it was created, and how to use the executable to create input data for the template. In addition, this report will also discuss the limitations for this template. Excerpts of the template code will be included within the main body of the report. An entire copy of the code is included in the appendix.
2 BACKGROUND

2.1 Moment Frames

Moment frames are a very common type of seismic force resisting system for steel buildings. Figure 1 is an example of a moment frame being used in a steel framed building. The special detailing around the beam to column connections are necessary to maintain a rigid connection. The rigid connections between the beams and columns provide lateral stiffness. In addition, the connections include a fuse of some kind, to allow the designer to control where and how failure will occur within the building. Figure 2 shows one example of an acceptable beam to column connection for special moment frames. This particular connection takes advantage of a reduced beam section, which controls the location of plastic deformation in the beam. Plastic deformation too close to the connection can cause connection failure. The template described in this report assumes a reduced beam section (Hamburger 2009).

In addition to clearly showing a reduced beam section, Figure 2 shows a good example of a moment connection. Some basic elements of a moment connection are the panel zone, continuity plates, and beam to column connections. Continuity plates are plates welded to the column to “continue” the beam flange from one beam to the other. The panel zone is the area of the column web, bounded by the continuity plates. Panel zones are the rectangular area the column where the beam would intersect the column. The connections between beam and column must be strong enough to ensure that the connection will remain ductile (Hamburger 2009).
2.2 Computer Models

Computer modeling is an important facet of structural analysis and design. As the computational power of the tools available to structural engineers improves, more detailed (and accurate) models can be created. For moment frame design, computer models are helpful in determining the lateral stiffness of moment frames. Typical moment frames are highly indeterminate and cumbersome to analyze precisely by hand.
2.3 OpenSees

Dynamic analyses are more complicated to run than static, and often require different software. This is due to the complexity and multitude of factors that can have significant effects when inertial forces are considered. The University of California Berkeley has developed an open source program called OpenSees, which stands for Open System for Earthquake Engineering Simulation. This program allows the user to create and run static and dynamic models by defining a structure and loading condition (University of California 2000).

OpenSees is a very powerful tool for dynamic analysis, but it is also very challenging to use. One challenge is that OpenSees has no graphical interface. All input is typed directly into the program as a text prompt, or it is sourced from a previously prepared document. There is no way to “undo” any input, so it is typical to prepare a series of commands into a text file and source that file into the program. This allows the user to edit the file later if there are errors (which is almost unavoidable) (University of California 2000).

2.4 The Parallelogram Model

The most typical model used in defining the geometry of frames is the point and centerline model. As the name implies, members of the structure are represented by lines that run along where the center of the member would be and are connected to other members at points. In steel design, this model is used almost exclusively. This model is very accurate along the mid-span section of members, but it becomes less accurate near the ends of the members. If one wished to observe deformations within the panel zone of a moment frame, it would be useless because that region would have been condensed down to a point.

The parallelogram model provides a solution to some of the challenges of the point and centerline model. In the parallelogram model, connections between beams and columns are
represented with a semi-rigid rectangle instead of a single point. Figure 3 shows representations of each model at a connection point. The parallelogram model simulates the panel zone by creating a rectangle of rigid elements that are pinned together at three corners and connected with a rotational spring at the fourth corner. The properties of the spring can be adjusted to account for the material properties of the system, and the dimensions of the panel zone are determined by the depth of the beams and columns (Hamburger et al. 2009).

![Figure 3: Point and Centerline Model vs. Parallelogram Model](image)

Figure 3: Point and Centerline Model vs. Parallelogram Model
3 TEMPLATE CREATION

A template was written with Tool Script Language (TCL) to help build a moment frame template in OpenSees incorporating the parallelogram model for the panel zones. Two main text files have been created. The first text file contains all of the commands to create the elements and connections. Also within this file are the commands to run analysis and define what information will be output. This file is written like a function where input is necessary. The second text file is for input values as well as the command for the function in the first file.

3.1 Preparation for Defining Elements

Within the code, two steps must be taken before elements can be defined. First, the order of analysis to be used must be defined (i.e. linear, p-delta, co-rotational). This is done separately for beams, columns, and for the rigid elements in the panel zone. Figure 4 displays an example of the analysis type for columns.

```tcl
set columnTransfTag [expr $template* 100 + 0]; #0 is for columns
if {$columnTransfCode==0} {
    geomTransf Linear $columnTransfTag;
} elseif {$columnTransfCode==1} {
    geomTransf PDelta $columnTransfTag;
} else {
    geomTransf Corotational $columnTransfTag;
}
```

Figure 4: Code Defining Analysis Type for Columns
The second step is setting up a coordinate system and defining the points where the elements will begin and terminate. These points and their respective coordinate values are defined based on given story heights and bay widths as well as beam and column dimensions. The general method for accomplishing this is to loop each bay or story and assign each point individually. An example of this for the horizontal coordinates can be seen in Figure 5.

```
set xgridLocations 0
    set cumLocation 0
    foreach width $bayWidths {
        set cumLocation [expr $cumLocation + $beamOffset]
        lappend xgridLocations $cumLocation
        set cumLocation [expr $cumLocation + $beamOffset]
        lappend xgridLocations $cumLocation
        set cumLocation [expr $cumLocation + $width - 2*$beamOffset]
        lappend xgridLocations $cumLocation
        set cumLocation [expr $cumLocation + $beamOffset]
        lappend xgridLocations $cumLocation
    }
    set cumLocation [expr $cumLocation + $beamOffset]
    lappend xgridLocations $cumLocation
```

Figure 5: Code Defining Horizontal Locations for the Template

3.2 Defining Elements

In OpenSees, elements are defined in physical space by defining the location of the endpoints. Various spatial and material properties are also assigned to an element depending on the element type. The endpoints of the elements are defined ahead-of-time as nodes. While several elements could share a node as an endpoint, this template defines a unique pair of nodes for each element. This helps avoid problems later when defining connections.
A specific naming convention has been used to help identify where nodes are and what they are connected to. The same convention is used on most OpenSees templates created at Brigham Young University. Each node is named with a ten digit number that can tell the user the location of the node, what the node is used for, and which frame number the node is in if multiple frames are paired together for analysis. The first two numbers indicate the template number. Each template is assigned a specific template number. Keeping this information in the names of nodes is useful if multiple templates are used within the same analysis. The next two numbers of the ten digit number refer to the frame number. This would mean up to 100 frames could be combined into one analysis while still maintaining this convention. The following two digits refer to the plane number. Nodes are assigned to different planes depending on how they will be used. Plane 0 is reserved for base nodes. These nodes are unmovable in the template and the rest of the template is connected to these nodes. Planes 1 and 2 are for the bottom and top of column elements. Planes 3 and 4 are for the left and right ends of beam elements. Planes 5 and 6 are for the ends of the elements in the panel zones. Plane 10 is not used for nodes, but for spring elements. Because these elements only exist at a point in the template they follow the naming convention for nodes. The last two sets of two numbers in the node name refer to the location of the node. The template is broken up into a grid in the x and y direction. Grid lines are not necessarily evenly spaced, but are located at points where nodes are located. The naming convention refers to the node’s location within this grid as opposed to a distance based number. The first pair of numbers in these sets signifies the y grid location and the second pair signifies the x grid location.

By using several layers of loops, the user is able have a great deal of control over the details of the template. The user can define the number of stories and bays as well as the size of
each beam and column shape individually. Within the main function, the code has a separate loop for columns, beams, and panel zones. For each run of each loop, two nodes are defined, and then an element is defined based on the two nodes. The procedure has a loop first through the stories (bottom to top) and an interior loop defining elements left to right. Figure 6 provides an example of this system for beams. As can also be seen in Figure 6, the properties of the element are re-defined with each iteration of the loop, thus allowing elements to be different at each location. The lists used to call the element properties are defined in the input text file.

The naming convention for elements is very similar to that of the nodes. Elements are named using a 10 digit number where the numbers have the same meaning as the nodes. For elements, plane 1 is for columns, plane 3 is for beams, plane 5 is for horizontal panel zone elements, and plane 6 is for vertical panel zone elements. Elements exist along a line as opposed to nodes, which exist at a point. To avoid ambiguity a convention is in place to define which x and y grid locations are used to define an element. For vertical elements, the top y grid location is used to name the element while all points along the element share the same x grid location. For horizontal elements, y grid locations are all the same and the left most x grid location is used to name the element.
for {set ygrid 2} {$ygrid <= [expr $numStories*3-1]} {incr ygrid 3} {
    for {set xgrid 2} {$xgrid <= [expr $numBays*3]} {incr xgrid 3} {
        set beamProps [lindex $beamPropsCombined [expr ($ygrid-2)/3]]
        set beamAreas [lindex $beamProps 0]
        set beamIs [lindex $beamProps 1]

        # Left node
        set plane 3
        set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
        set X [lindex $xgridLocations [expr $xgrid]]
        set Y [lindex $ygridElevations [expr $ygrid]]
        node $nodeI $X $Y
        puts "node $nodeI $X $Y"

        # Right node
        set plane 4
        set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid+1]
        set X [lindex $xgridLocations [expr $xgrid+1]]
        set Y [lindex $ygridElevations [expr $ygrid]]
        node $nodeJ $X $Y
        puts "node $nodeJ $X $Y"

        # Beam
        set plane 3
        set elemID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
        set area [lindex $beamAreas [expr ($xgrid-2)/3]]
        set I [lind $beamIs [expr ($xgrid-2)/3]]
        element elasticBeamColumn $elemID $nodeI $nodeJ $area $Es $I $beamTransfTag
        puts "element elasticBeamColumn $elemID $nodeI $nodeJ $area $Es $I $beamTransfTag"
    }
}

Figure 6: Example of Code Defining Nodes and Elements
3.3 Defining Connections

After all of the elements have been defined, they are connected together. In OpenSees, nodes can be connected together through a slave and master system. In other words, one node (slave) is forced to have the same displacements and/or rotations as the other node (master). If the connection is pinned, only displacements are passed to the slave node. If the connection is fixed, rotations are passed from the master node’s element to the slave node’s element as well as the displacement. Rotational springs are also used at certain points to connect elements. The primary spot where rotational springs are used is in one of the corners of the panel zone as required for the parallelogram model. Rotational springs are also used to connect elements to the panel zone. These springs are included to represent the effect of plastic hinges forming. Figure 7 is a copy of the sub-function that defines the panel zone springs.
proc rotPanelZone2D {eleID nodeR nodeC E Fy dc bf_c tf_c tp db Ry as} {  
    # Trilinear Spring  
    # Yield Shear  
    set Vy [expr 0.55 * $Fy * $dc * $tp];  
    # Shear Modulus  
    set G [expr $E/(2.0 * (1.0 + 0.30))]
    # Elastic Stiffness
    set Ke [expr 0.95 * $G * $tp * $dc];
    # Plastic Stiffness
    set Kp [expr 0.95 * $G * $bf_c * ($tf_c * $tf_c) / $db];
    # Define Trilinear Equivalent Rotational Spring
    # Yield point for Trilinear Spring at gamma1_y
    set gamma1_y [expr $Vy/$Ke]; set M1y [expr $gamma1_y * ($Ke * $db)];
    # Second Point for Trilinear Spring at 4 * gamma1_y
    set gamma2_y [expr 4.0 * $gamma1_y]; set M2y [expr $M1y + ($Kp * $db) *\   
            ($gamma2_y - $gamma1_y)];
    # Third Point for Trilinear Spring at 100 * gamma1_y
    set gamma3_y [expr 100.0 * $gamma1_y]; set M3y [expr $M2y + ($as * $Ke *\   
            $db) * ($gamma3_y - $gamma2_y)];
    # Hysteretic Material without pinching and damage (same mat ID as Ele ID)
    uniaxialMaterial Hysteretic $eleID $M1y $gamma1_y $M2y $gamma2_y $M3y\   
                $gamma3_y [expr -$M1y] [expr -$gamma1_y] [expr -$M2y] [expr -$\   
                $gamma2_y] [expr -$M3y] [expr -$gamma3_y] 1 1 0.0 0.0 0.0
    element zeroLength $eleID $nodeR $nodeC -mat $eleID -dir 6
    equalDOF $nodeR $nodeC 1 2
    # Constrain the translational DOF with a multi-point constraint
    # Left Top Corner of PZ
    set nodeR_1 [expr $nodeR - 2];
    set nodeR_2 [expr $nodeR_1 + 1];
    # Right Bottom Corner of PZ
    set nodeR_6 [expr $nodeR + 3];
    set nodeR_7 [expr $nodeR_6 + 1];
    # Left Bottom Corner of PZ
    set nodeL_8 [expr $nodeR + 5];
    set nodeL_9 [expr $nodeL_8 + 1];
    # retained constrained DOF_1 DOF_2
    #equalDOF $nodeR_1 $nodeR_2 1 2
    #equalDOF $nodeR_6 $nodeR_7 1 2
    #equalDOF $nodeL_8 $nodeL_9 1 2}

Figure 7: Sub-function Defining Panel Zone Springs
The verification of this template is based on the lateral deflections of a single story one bay frame under a pushover analysis. Equations 1 through 3 provide three separate components of story drift, which are story drift to beam flexure, column flexure, and panel zone shear deformation. The sum of these components are the total theoretical deflection. These calculations then can be used to compare to the output of the OpenSees template. Figure 8 provides a visual example of the contributions to story drift from equations 4.1 through 4.3 (Hamburger et al. 2009). Many tests were performed in the OpenSees template using different beam and column shapes, story heights, and bay widths. Table 1 compares data output from the OpenSees template and the theoretical model. While some configurations of the variables were closer to the theoretical model than others, almost all were within 10% accuracy. The cases where the template and model were further apart were cases in which the bay widths or story heights were much smaller than would be realistic in a full scale structure (distances of 5 or 10 ft.).

\[
\delta_r = \frac{h^2(1-2dc)}{6E}\frac{\left(\frac{I_1}{l_1-\Delta c} + \frac{I_2}{l_2-\Delta c}\right)}{V_{col}} \\
(4.1)
\]

\[
\delta_c = \left(\frac{h-d_b}{12E}\right)^3 V_{col} \\
(4.2)
\]
\[
\delta_p = \frac{(h-d_b)\left(\frac{h}{d_b} - 1\right)}{Gt_d d_c} V_{col}
\]  

This paragraph will define the variables in equations 4.1 through 4.3. \(\delta_b\) is the story drift due to beam flexure. \(\delta_c\) is the story drift due to column flexure. \(\delta_p\) is the story drift due to panel zone shear deflection. \(V_{col}\) is the column shear force. \(h\) is the story height (centerline dimension). \(l_1\) and \(l_2\) are the beam spans on either side of the panel zone (centerline dimension). \(I_1\) and \(I_2\) are the moments of inertia of each beam. \(I_c\) is the moment of inertia of the column. \(d_b\) is the depth of the beams. \(d_c\) is the depth of the column. \(t_p\) is the thickness of the panel zone plate.

Figure 8: Different Contributions to Story Drift (Haburger et al. 2009)
<table>
<thead>
<tr>
<th>Story Height X Bay Width</th>
<th>Open Sees Template (Stiffness in k/in)</th>
<th>Theoretical Model (Stiffness in k/in)</th>
<th>Percent Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W12X136 W8X40 W21X166 W16X100 W10X26 W14X342</td>
<td>W12X136 W8X40 W21X166 W16X100 W10X26 W14X342</td>
<td></td>
</tr>
<tr>
<td>5X10</td>
<td>504.6556 52.8783 1575.61</td>
<td>615.368 59.825 1832</td>
<td>-18% -12% -14%</td>
</tr>
<tr>
<td>10X10</td>
<td>84.3258 9.23467 257.225</td>
<td>86.627 9.42 259.643</td>
<td>-3% -2% -1%</td>
</tr>
<tr>
<td>14X10</td>
<td>33.7318 3.76849 103.575</td>
<td>33.846 3.781 103.25</td>
<td>0% 0% 0%</td>
</tr>
<tr>
<td>18X10</td>
<td>16.7681 1.89724 51.8916</td>
<td>16.715 1.893 51.762</td>
<td>0% 0% 0%</td>
</tr>
<tr>
<td>20X10</td>
<td>12.4696 1.41756 38.7279</td>
<td>12.417 1.413 38.7</td>
<td>0% 0% 0%</td>
</tr>
<tr>
<td>25X10</td>
<td>6.62345 0.759706 20.7311</td>
<td>6.595 0.756 20.832</td>
<td>0% 0% 0%</td>
</tr>
<tr>
<td>30X10</td>
<td>3.9703 0.453703 12.3801</td>
<td>3.92 0.452 12.511</td>
<td>1% 0% -1%</td>
</tr>
<tr>
<td>5X15</td>
<td>387.4861 41.2959 1189.25</td>
<td>517.613 49.218 1567</td>
<td>-25% -16% -24%</td>
</tr>
<tr>
<td>10X15</td>
<td>72.0626 7.84302 218.51</td>
<td>77.028 8.2 233.267</td>
<td>-6% -4% -6%</td>
</tr>
<tr>
<td>14X15</td>
<td>29.9813 3.31364 91.8485</td>
<td>30.779 3.376 94.54</td>
<td>-3% -2% -3%</td>
</tr>
<tr>
<td>18X15</td>
<td>15.2716 1.70661 47.2849</td>
<td>15.43 1.72 48.007</td>
<td>-1% -1% -2%</td>
</tr>
<tr>
<td>20X15</td>
<td>11.46 1.28625 35.6358</td>
<td>11.529 1.292 36.073</td>
<td>-1% 0% -1%</td>
</tr>
<tr>
<td>25X15</td>
<td>6.19207 0.700958 19.4642</td>
<td>6.194 0.701 19.613</td>
<td>0% 0% -1%</td>
</tr>
<tr>
<td>30X15</td>
<td>3.71933 0.423684 11.7883</td>
<td>3.712 0.423 11.866</td>
<td>0% 0% -1%</td>
</tr>
<tr>
<td>5X20</td>
<td>315.7694 33.9453 966.57</td>
<td>446.658 41.806 1369</td>
<td>-29% -19% -29%</td>
</tr>
<tr>
<td>10X20</td>
<td>62.9943 6.82155 190.912</td>
<td>69.344 7.26 211.756</td>
<td>-9% -6% -10%</td>
</tr>
<tr>
<td>14X20</td>
<td>26.9694 2.95713 82.5697</td>
<td>28.222 3.049 87.185</td>
<td>-4% -3% -5%</td>
</tr>
<tr>
<td>18X20</td>
<td>13.9964 1.55016 43.3151</td>
<td>14.328 1.575 44.759</td>
<td>-2% -2% -3%</td>
</tr>
<tr>
<td>20X20</td>
<td>10.5781 1.17647 32.8985</td>
<td>10.76 1.191 33.78</td>
<td>-2% -1% -3%</td>
</tr>
<tr>
<td>25X20</td>
<td>5.79458 0.64996 18.2166</td>
<td>5.838 0.653 18.528</td>
<td>-1% 0% -2%</td>
</tr>
<tr>
<td>30X20</td>
<td>3.51514 0.396844 11.148</td>
<td>3.525 0.398 11.285</td>
<td>0% 0% -1%</td>
</tr>
<tr>
<td>5X25</td>
<td>266.9528 28.8404 818.774</td>
<td>392.811 36.335 1215</td>
<td>-32% -21% -33%</td>
</tr>
<tr>
<td>10X25</td>
<td>56.01347 6.03852 170.129</td>
<td>63.054 6.513 193.887</td>
<td>-11% -7% -12%</td>
</tr>
<tr>
<td>14X25</td>
<td>24.5202 2.67063 75.1618</td>
<td>26.057 2.78 80.891</td>
<td>-6% -4% -7%</td>
</tr>
<tr>
<td>18X25</td>
<td>12.919 1.42015 40.004</td>
<td>13.374 1.454 41.923</td>
<td>-3% -2% -5%</td>
</tr>
<tr>
<td>20X25</td>
<td>9.8216 1.08401 30.5573</td>
<td>10.087 1.104 31.762</td>
<td>-3% -2% -4%</td>
</tr>
<tr>
<td>25X25</td>
<td>5.4427 0.605818 17.1126</td>
<td>5.521 0.612 17.557</td>
<td>-1% -1% -3%</td>
</tr>
<tr>
<td>30X25</td>
<td>3.32987 0.373122 10.2642</td>
<td>3.356 0.375 10.758</td>
<td>-1% -1% -5%</td>
</tr>
<tr>
<td>5X30</td>
<td>231.423 25.0804 712.225</td>
<td>350.551 32.129 1092</td>
<td>-34% -22% -35%</td>
</tr>
<tr>
<td>10X30</td>
<td>50.4578 5.41824 153.753</td>
<td>57.81 5.906 178.782</td>
<td>-13% -8% -14%</td>
</tr>
<tr>
<td>14X30</td>
<td>22.4882 2.43521 69.0795</td>
<td>24.201 2.554 75.446</td>
<td>-7% -5% -8%</td>
</tr>
<tr>
<td>18X30</td>
<td>11.9984 1.31042 37.2021</td>
<td>12.538 1.349 39.425</td>
<td>-4% -3% -6%</td>
</tr>
<tr>
<td>20X30</td>
<td>9.16745 1.00511 28.5511</td>
<td>9.493 1.029 29.97</td>
<td>-3% -2% -5%</td>
</tr>
<tr>
<td>25X30</td>
<td>5.13113 0.567306 16.1412</td>
<td>5.237 0.575 16.683</td>
<td>-2% -1% -3%</td>
</tr>
<tr>
<td>30X30</td>
<td>3.19278 0.352067 10.0335</td>
<td>3.202 0.355 10.278</td>
<td>0% -1% -2%</td>
</tr>
</tbody>
</table>
5 USING THE INPUT GENERATOR

Because the syntax for the input file can be challenging to use without mistakes, a tool has been created to facilitate creating the OpenSees input file. This tool has been developed within a spreadsheet program created previously by Alex Hawkins. The spreadsheet’s primary function is to suggest beam and column shapes for moment frames based on frame size and loading conditions. What has been added to this program is a button that takes the parameters used to pick member shapes as well as the member shapes and create an input file for OpenSees. It should be noted that the user may make changes to the beam and column size suggestions, thus giving the user total control over the input file. Creating the input file is very simple. After following the instructions on the spreadsheet to generate shapes for the members of the building, all one needs to do is press the Generate OS Input button located in the Building Parameters sheet. Figure 9 shows a screenshot of this sheet.

Figure 9: Screenshot of Moment Frame Designer with OS Input Button
6  TEMPLATE OUTPUT

There is a large list of information that OpenSees can output. To obtain output data from OpenSees, recorders must be defined prior to analysis. OpenSees can record data such as deflections, forces, and accelerations. OpenSees can also record information as it is at the end of an analysis, or it can record information throughout the analysis at specified time steps. Once the output data is recorded, it is printed. This data can be printed to the screen or sent to a text file. Unless there are only one or two pieces of data needed, it is usually best to send the template output to one or more text files (University of California 2000).

This template records, and prints lateral displacements and external forces in the same direction for each story. This information is adequate to obtain the overall stiffness of the building being analyzed. When used with a pushover analysis, one can observe how the stiffness of the building changes as plastic deformation begins to occur in the structure. Figure 10 shows the code for defining the recorder for one of the stories. The –file section of the code defines the location that the text file will be stored, and the –node section defines what will be recorded.

```
Recorder Node -file $dataDir/sixStoryDisp$frameNum$ygrid.out -node $nodeID -dof 1 -time disp
```

Figure 10: Code Defining Output Information
7 TEMPLATE LIMITATIONS

While this template is able to provide useful information to its users, there are some limitations to its use. The template is unable to function with rotational springs connecting the beam elements with the column panel zone boxes. The purpose of these springs is to be able to simulate the formation of plastic hinges. The template would be more versatile in its use if these springs were included. It should be noted that springs in the columns that fulfil a similar purpose do exist in the template. Secondly, all panel zones are defined to have the same depth in the template. Panel zone thicknesses are still unique to each individual panel zone, and based on the thickness of the column beneath the panel zone. In reality, the size of the panel zone is a function of the beam and column depths and the column thickness. Unless very different beam and column sizes are used within the same frame, this issue should not present a notable difference in the template.
CONCLUSIONS

The OpenSees computer template created for this report is based on the parallelogram model for panel zones. This model is more accurate for determining elastic stiffness and story drifts than the typical centerline and point model. This is due to the fact that the parallelogram model accounts for deformations in the panel zone. While there is some error between the computer template and the theoretical model, the computer template is accurate enough to be considered acceptable.
REFERENCES


University of California (2000). OpenSees. opensees.berkeley.edu (September 2014).
APPENDIX A. PARALELLOGRAM TEMPLATE OPENSEES CODE

source five.tcl
source LookupShapeProp.tcl
set filename "AISC_Database.csv"
set dataDir "Results"

set numFrames 1
set numStories 1
set numBays 1
set storyHeights [list [expr 14*12]]
set bayWidths [list [expr 30*12]]
set beamOffset [expr 12*1.5]
set columnOffset [expr 12*2]
set areaB 10000
set IB 10000
set baseFixity 0; #0 for columns pinned at base, 1 for fixed columns

set columnTransfCode 0; #0 is linear, 1 is P-Delta, 2 is co-rotational
set beamTransfCode 0; #0 is linear, 1 is P-Delta, 2 is co-rotational
set boxTransfCode 0; #0 is linear, 1 is P-Delta, 2 is co-rotational

set colShapes1 [list W12X136 W12X136]; #use upper-case X, should be number of bays + 1 column in the list.
set columnProps1 [LookupShapeProp $filename $colShapes1]
set colShapes2 [list W12X136 W12X136]
set columnProps2 [LookupShapeProp $filename $colShapes2]

set beamShapes1 [list W16X100]; #use upper-case X
set beamProps1 [LookupShapeProp $filename $beamShapes1]
set beamShapes2 [list W16X100];
set beamProps2 [LookupShapeProp $filename $beamShapes2]
set columnPropsCombined [list $columnProps1 $columnProps2]
set beamPropsCombined [list $beamProps1 $beamProps2]

set maxPush 0.1; # percent of frame height
set Es 29000
set Fy 50

set n 10; # stiffness multiplier for rotational spring
set McMy 1.02; # ratio of capping moment to yield moment, Mc / My

wipe all;

model BasicBuilder -ndm 2 -ndf 3

five $numFrames $numStories $numBays $beamOffset $columnOffset $storyHeights $bayWidths $Es $Fy $baseFixity $columnPropsCombined $beamPropsCombined $areaB $IB $columnTransfCode $beamTransfCode $boxTransfCode $n $McMy

pushover $numFrames $numStories $storyHeights $maxPush $dataDir
wipe all;

proc five {numFrames numStories numBays beamOffset columnOffset storyHeights bayWidths Es Fy baseFixity columnPropsCombined beamPropsCombined areaB IB columnTransfCode beamTransfCode boxTransfCode n McMy} {

set template 6

set columnTransfTag [expr $template* 100 + 0]; #0 is for columns
if {$columnTransfCode==0} {
    geomTransf Linear $columnTransfTag;
} elseif {$columnTransfCode==1} {
    geomTransf PDelta $columnTransfTag;
} else {
    geomTransf Corotational $columnTransfTag;
}

set beamTransfTag [expr $template* 100 + 1]; #1 is for beams
if {$beamTransfCode==0} {

if {$beamTransfCode==0} {
    geomTransf Linear $beamTransfTag;
} elseif {$beamTransfCode==1} {
    geomTransf PDelta $beamTransfTag;
} else {
    geomTransf Corotational $beamTransfTag;
}

set boxTransfTag [expr $template* 100 + 2]; #1 is for beams
if {$boxTransfCode==0} {
    geomTransf Linear $boxTransfTag;
} elseif {$boxTransfCode==1} {
    geomTransf PDelta $boxTransfTag;
} else {
    geomTransf Corotational $boxTransfTag;
}

set ygridElevations 0
set cumElevation 0
foreach height $storyHeights {
    if {$cumElevation > 0} {
        set cumElevation [expr $cumElevation + $height - 2*$columnOffset]
        lappend ygridElevations $cumElevation
    } else {
        set cumElevation [expr $cumElevation + $height - $columnOffset]
        lappend ygridElevations $cumElevation
    }
}

set xgridLocations 0
set cumLocation 0
foreach width $bayWidths {
    set cumLocation [expr $cumLocation + $beamOffset]
    lappend xgridLocations $cumLocation
    set cumLocation [expr $cumLocation + $beamOffset]
    lappend xgridLocations $cumLocation
    set cumLocation [expr $cumLocation + $beamOffset]
    lappend xgridLocations $cumLocation
}

set cumLocation [expr $cumLocation + $beamOffset]
lappend xgridLocations $cumLocation

for { set frameNum 1 } { $frameNum <= $numFrames } { incr frameNum } {

   # base nodes

   set plane 0
   for { set xgrid 1 } { $xgrid <= [expr $numBays*3 + 1] } { incr xgrid 3 } {
      set ygrid 0
      set nodeID [expr $template*10000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
      set X [lindex $xgridLocations [expr $xgrid]]
      set Y [lindex $ygridElevations [expr $ygrid]]
      node $nodeID $X $Y
      puts "node $nodeID $X $Y"
   }
   puts "base nodes defined"

   # columns

   # First column
   for { set xgrid 1 } { $xgrid <= [expr $numBays*3 + 1] } { incr xgrid 3 } {

      set ygrid 0
      set columnProps [lindex $columnPropsCombined $ygrid]
      set columnAreas [lindex $columnProps 0]
      set columnIs [lindex $columnProps 1]
      set plane 1
      set nodeI [expr $template*10000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
      set X [lindex $xgridLocations [expr $xgrid]]
      set Y [lindex $ygridElevations [expr $ygrid]]
      node $nodeI $X $Y
      puts "node $nodeI $X $Y"
      set plane 2
      set nodeJ [expr $template*10000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
      set X [lindex $xgridLocations [expr $xgrid]]
      set Y [lindex $ygridElevations [expr $ygrid]]
      node $nodeJ $X $Y
      puts "node $nodeJ $X $Y"

   }
```python
$plane*10000 + ($ygrid+1)*100 +
$frameNum*1000000\$xgrid] + $plane*10000 + ($ygrid+1)*100 +

set X [lindex $xgridLocations [expr $xgrid]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Column
set elemID [expr $template*100000000 +
+ $plane*10000 + ($ygrid+1)*100 +
$frameNum*1000000\$xgrid]

set area [lindex $columnAreas [expr ($xgrid-1)/3]]
set I [lindex $columnIs [expr ($xgrid-1)/3]]
element elasticBeamColumn $elemID $nodeI $nodeJ $area
$Es \$Es $I $columnTransfTag"puts "element elasticBeamColumn $elemID $nodeI $nodeJ

$columnTransfTag
$area \

$Es $I $columnTransfTag"
}

# All other columns
for {set ygrid 3} {$ygrid <= [expr $numStories*3 - 1]} {incr ygrid 3} {
    for {set xgrid 1} {$xgrid <= [expr $numBays*3 + 1]} {incr xgrid 3} {
        set columnProps [lindex $columnPropsCombined [expr
$ygrid/3]]
        set columnAreas [lindex $columnProps 0]
        set columnIs [lindex $columnProps 1]

        # Bottom node
        set plane 1
        set nodeI [expr $template*100000000 +
+ $plane*10000 + ($ygrid+1)*100 +
$frameNum*1000000\$xgrid]

        set X [lindex $xgridLocations [expr $xgrid]]
        set Y [lindex $ygridElevations [expr $ygrid]]
        node $nodeI $X $Y
        puts "node $nodeI $X $Y"
```
# Top node
set plane 2
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+2)*100 + $xgrid]
set X [lindex $xgridLocations [expr $xgrid]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Column
set plane 1
set elemID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid]
set area [lindex $columnAreas [expr ($xgrid-1)/3]]
set I [lindex $columnIs [expr ($xgrid-1)/3]]

element elasticBeamColumn $elemID $nodeI $nodeJ $area $Es $I $columnTransfTag
puts "element elasticBeamColumn $elemID $nodeI $nodeJ $Es $I $columnTransfTag"

puts "column nodes and elements defined"

# beams

for {set ygrid 2} {$ygrid <= [expr $numStories*3-1]} {incr ygrid 3} {
    for {set xgrid 2} {$xgrid <= [expr $numBays*3]} {incr xgrid 3} {
        set beamProps [lindex $beamPropsCombined [expr ($ygrid-2)/3]]
        set beamAreas [lindex $beamProps 0]
        set beamIs [lindex $beamProps 1]

        # Left node
        set plane 3
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]

set X [lindex $xgridLocations [expr $xgrid]]
set Y [lindex $ygridElevations [expr $ygrid]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# Right node
set plane 4
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid+1]
set X [lindex $xgridLocations [expr $xgrid+1]]
set Y [lindex $ygridElevations [expr $ygrid]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 3
set elemID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
set area [lindex $beamAreas [expr ($xgrid-2)/3]]
set I [lindex $beamIs [expr ($xgrid-2)/3]]

for {set ygrid 2} {$ygrid <= [expr $numStories*3]} {incr ygrid 3} {
    for {set xgrid 1} {$xgrid <= [expr $numBays*3+1]} {incr xgrid 3} {

puts "beam nodes and elements defined"
} }
# Section 1

# NodeI
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid - 1]
set X [lindex $xgridLocations [expr $xgrid-1]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# NodeJ
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid]
set X [lindex $xgridLocations [expr $xgrid]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 5
set elemID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid-1]
element elasticBeamColumn $elemID $nodeI $nodeJ
$areaB $Es $IB
$boxTransfTag
puts "element elasticBeamColumn $elemID $nodeI $nodeJ
$areaB $Es $IB $beamTransfTag"

# Section 2

# NodeI
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid-1]
$xgrid]

set X [lindex $xgridLocations [expr $xgrid]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# NodeJ
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid+1]
set X [lindex $xgridLocations [expr $xgrid+1]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 5
set elemID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid]

$areaB $Es \n$boxTransfTag
$areaB \n
$Es $IB $beamTransfTag"

# Section 3

# NodeI
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid+1]
set X [lindex $xgridLocations [expr $xgrid+1]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"
# NodeJ
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid)*100 + $xgrid+1]
set X [lindex $xgridLocations [expr $xgrid+1]]
set Y [lindex $ygridElevations [expr $ygrid]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 6
set elemID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid)*100 + $xgrid+1]
element elasticBeamColumn $elemID $nodeI $nodeJ $areaB $Es $IB $boxTransfTag
puts "element elasticBeamColumn $elemID $nodeI $nodeJ $areaB $Es $IB $beamTransfTag"

# Section 4
# NodeI
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid)*100 + $xgrid+1]
set X [lindex $xgridLocations [expr $xgrid+1]]
set Y [lindex $ygridElevations [expr $ygrid]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# NodeJ
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid+1]
set X [lindex $xgridLocations [expr $xgrid+1]]
set Y [lindex $ygridElevations [expr $ygrid-1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 6
set elemID [expr $template*100000000 +
$frameNum*1000000 \
$plane*10000 + ($ygrid-1)*100 +
$xgrid+1]
element elasticBeamColumn $elemID $nodeI $nodeJ
$areaB $Es \ 
$boxTransfTag
puts "element elasticBeamColumn $elemID $nodeI $nodeJ
$areaB $Es $Ib $beamTransfTag"

# Section 5

# NodeI
set plane 5
set nodeI [expr $template*100000000 +
$frameNum*1000000 \n$plane*10000 + ($ygrid-1)*100 +
$xgrid+1]
set X [lindex $xgridLocations [expr $xgrid+1]]
set Y [lindex $ygridElevations [expr $ygrid-1]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# NodeJ
set plane 6
set nodeJ [expr $template*100000000 +
$frameNum*1000000 \n$plane*10000 + ($ygrid-1)*100 +
$xgrid]
set X [lindex $xgridLocations [expr $xgrid]]
set Y [lindex $ygridElevations [expr $ygrid-1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 5
set elemID [expr $template*100000000 +
$frameNum*1000000 \n$areaB $Es $Ib $beamTransfTag"
puts "element elasticBeamColumn $elemID $nodeI $nodeJ $areaB $Es $IB $boxTransfTag"

# Section 6

# NodeI
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid]
set X [lindex $xgridLocations [expr $xgrid]]
set Y [lindex $ygridElevations [expr $ygrid-1]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# NodeJ
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid-1]
set X [lindex $xgridLocations [expr $xgrid-1]]
set Y [lindex $ygridElevations [expr $ygrid-1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 5
set elemID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid-1]
element elasticBeamColumn $elemID $nodeI $nodeJ $areaB $Es $IB $boxTransfTag"
puts "element elasticBeamColumn $elemID $nodeI $nodeJ
$areaB \n$Es $IB $beamTransfTag"

# Section 7

# NodeI
set plane 5
set nodeI [expr $template*100000000 +
$frameNum*1000000 \n$plane*10000 + ($ygrid-1)*100 +
$xgrid-1] set X [lindex $xgridLocations [expr $xgrid-1]]
set Y [lindex $ygridElevations [expr $ygrid-1]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# NodeJ
set plane 6
set nodeJ [expr $template*100000000 +
$frameNum*1000000 \n$plane*10000 + (ygrid)*100 +
$xgrid-1] set X [lindex $xgridLocations [expr $xgrid-1]]
set Y [lindex $ygridElevations [expr $ygrid]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 6
set elemID [expr $template*100000000 +
$frameNum*1000000 \n$plane*10000 + ($ygrid-1)*100 +
$xgrid-1] element elasticBeamColumn $elemID $nodeI $nodeJ
$areaB $Es \n$boxTransfTag $IB
puts "element elasticBeamColumn $elemID $nodeI $nodeJ
$areaB \n$Es $IB $beamTransfTag"

# Section 8
# NodeI
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid)*100 + $xgrid-1]
set X [lindex $xgridLocations [expr $xgrid-1]]
set Y [lindex $ygridElevations [expr $ygrid]]
node $nodeI $X $Y
puts "node $nodeI $X $Y"

# NodeJ
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid-1]
set X [lindex $xgridLocations [expr $xgrid-1]]
set Y [lindex $ygridElevations [expr $ygrid+1]]
node $nodeJ $X $Y
puts "node $nodeJ $X $Y"

# Beam
set plane 6
set elemID [expr $template*100000000 + $plane*10000 + ($ygrid)*100 + $xgrid-1]
set elemID [expr $elemID + 1]
element elasticBeamColumn $elemID $nodeI $nodeJ $areaB $Es $IB
puts "element elasticBeamColumn $elemID $nodeI $nodeJ $areaB $Es $IB $beamTransfTag"
}]
puts "box nodes and elements defined"

# connections

# base node connections
for {set xgrid 1} {$xgrid <= [expr $numBays*3+1]} {incr xgrid 3} {
    set ygrid 0
    set plane 0
}
set nodeID [expr $template*100000000 + $frameNum*1000000 \ + $plane*10000 + $ygrid*100 + $xgrid]  
fix $nodeID 1 1 1
puts "$nodeID fixed in space"
}
puts "base node connections defined"

# Column connections

# Bottom columns
for {set xgrid 1} {$xgrid <= [expr $numBays*3+1]} {incr xgrid 3} {
    set ygrid 0
    # Base to bottom
    set plane 1
    set nodeJ [expr $template*100000000 + $frameNum*1000000 \ + $plane*10000 + $ygrid*100 + $xgrid]
    set plane 0
    set nodeI [expr $template*100000000 + $frameNum*1000000 \ + $plane*10000 + $ygrid*100 + $xgrid]
    if {$baseFixity==0} {
        equalDOF $nodeI $nodeJ 1 2
        puts "$nodeJ pinned to $nodeI"
    } else {
        equalDOF $nodeI $nodeJ 1 2 3
        puts "$nodeJ fixed to $nodeI"
    }
    # top of column
    set plane 6
    set nodeI [expr $template*100000000 + $frameNum*1000000 \ + $plane*10000 + ($ygrid+1)*100 + $xgrid]
    set plane 2
    set nodeJ [expr $template*100000000 + $frameNum*1000000 \ + $plane*10000 + ($ygrid+1)*100 + $xgrid]
    rigidLink beam $nodeI $nodeJ
    puts "$nodeI $nodeJ fixed"
}

# Other columns
for {set ygrid 2} {$ygrid <= [expr ($numStories-1)*3-1]} {incr $ygrid 3} {
    for {set xgrid 1} {$xgrid <= [expr $numBays*3+1]} {incr xgrid 3} {

# Bottom to box
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid*100 + $plane*10000 + $xgrid]
set plane 1
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid*100 + $plane*10000 + ($ygrid+1)*100 + $xgrid]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Top to box
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
set plane 2
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

puts "column connections defined"

# beam connections
for {set ygrid 2} {$ygrid <= [expr $numStories*3]}  {incr ygrid 3} {
    for {set xgrid 1} {$xgrid <= [expr $numBays*3]} {incr xgrid 3} {
        # left beam to box
        set plane 6
        set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid+1]
        set plane 3
        set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid+1]
        rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Right beam to box
set plane 6
set nodeI \[expr $template*100000000 +

\$frameNum*1000000 \n+ $plane*10000 + $ygrid*100 +
$\xgrid+2] \n
set plane 4
set nodeJ \[expr $template*100000000 +

\$frameNum*1000000 \n+ $plane*100000000 + $plane*10000 + $ygrid*100 +
$\xgrid+2] \n
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

puts "beam connections defined"

# Box connections

for  {set ygrid 2} {$ygrid <= [expr $numStories*3]} {incr ygrid 3} {
for {set xgrid 1} {$xgrid <= [expr $numBays*3+1]} {incr xgrid 3} {

# Point 1
set plane 5
set nodeI \[expr $template*100000000 +

\$frameNum*1000000 \n+ $plane*10000 + ($ygrid+1)*100 +
$\xgrid-1] \n
set plane 6
set nodeJ \[expr $template*100000000 +

\$frameNum*1000000 \n+ $plane*10000 + ($ygrid+1)*100 +
$\xgrid-1] \n
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Point 2
set plane 5
set nodeI \[expr $template*100000000 +

\$frameNum*1000000 \n+ $plane*10000 + ($ygrid+1)*100 +
$\xgrid] \n
set plane 6
set nodeJ [expr $template*100000000 + $plane*10000 + ($ygrid+1)*100 + $xgrid]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Point 3
set plane 5
set nodeI [expr $template*100000000 + $plane*1000000 + $xgrid]
set nodeJ [expr $template*100000000 + $plane*1000000 + $xgrid+1]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Point 4
set plane 5
set nodeI [expr $template*100000000 + $plane*1000000 + $xgrid+1]
set nodeJ [expr $template*100000000 + $plane*1000000 + $xgrid]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Point 5
set plane 5
set nodeI [expr $template*100000000 + $plane*1000000 + $xgrid+1]
set nodeJ [expr $template*100000000 + $plane*1000000 + $xgrid-1]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Point 6
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid]
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Point 7
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid-1]
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid-1)*100 + $xgrid-1]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

# Point 8
set plane 5
set nodeI [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid)*100 + $xgrid-1]
set plane 6
set nodeJ [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + ($ygrid)*100 + $xgrid-1]
rigidLink beam $nodeI $nodeJ
puts "$nodeI $nodeJ fixed"

}
puts "box connections defined"
}
puts "node and elements complete"
}

#Pushover
proc pushover {numFrames numStories storyHeights maxPush dataDir} {
    set template 6

    #Define the load pattern
    pattern Plain 200 Linear {
        #these lines compute a "cum" value that is used to divide story load
        #pattern such that the sum of all story load patterns will be one.
        set cum 0.0
        for {set ygrid 2} {$ygrid <= [expr $numStories*3-1]} {incr ygrid 3} {
            set cum [expr $cum + [expr ($ygrid+1)/3]]
            puts "cum is $cum"
        }
    }

    #only apply pushover to one frame - other dragged by constraints
    set frameNum 1
    set plane 6; #box end node
    for {set ygrid 2} {$ygrid <= [expr $numStories*3-1]} {incr ygrid 3} {
        set xgrid 0; #nodes on the left will get the lateral load
        set nodeID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
        load $nodeID [expr ($ygrid+1)/3/$cum] 0.0 0.0
    }
}

#Define the displacement recorders
set plane 6
set xgrid 2
set frameNum 1
for {set ygrid 2} {$ygrid <= [expr $numStories*3-1]} {incr ygrid 3} {
    set nodeID [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
    recorder Node -file $dataDir/fiveStoryDisp$frameNum$ygrid.out -node $nodeID -dof 1 -time disp
} #set the displacement control node plane
set plane 6

set totalHeight 0
foreach height $storyHeights {
    set totalHeight [expr $totalHeight + $height]
}

#Run the pushover analysis
set frameNum 1
set ygrid [expr $numStories*3-1]
set xgrid 2
set dispControlNode [expr $template*100000000 + $frameNum*1000000 + $plane*10000 + $ygrid*100 + $xgrid]
set dispControlDOF 1
set dispMax [expr $maxPush*$totalHeight]
set dispIncr [expr 0.01]

# analysis commands
constraints Plain; # how it handles boundary conditions
numberer RCM; # renumber dof's to minimize band-width\( \text{optimization} \)
    system BandGeneral; # how to store and solve the system of\( \text{equations in the analysis}\)

model: try UmfPack)
    test NormUnbalance 1.0e-6 400; # tolerance, max iterations
algorithm Newton; # use Newton’s solution algorithm:
updates tangent stiffness at every\( \text{iteration} \)
integrator DisplacementControl $dispControlNode $dispControlDOF $dispIncr # use displacement-controlled analysis
analysis Static;  # define type of analysis:

static for

pushover
    set Nsteps [expr int($dispMax/$dispIncr)]; # number of pushover analysis steps
    set ok [analyze $Nsteps];  # this will return zero if no convergence

problems were encountered

puts "pushover complete";  # display this message in the command window
}

proc rotPanelZone2D {eleID nodeR nodeC E Fy dc bf_c tf_c tp db Ry as} {

    # Trilinear Spring
    # Yield shear
    set Vy [expr 0.55 * $Fy * $dc * $tp];
    # Shear Modulus
    set G [expr $E/(2.0 * (1.0 + 0.30))]
    # Elastic Stiffness
    set Ke [expr 0.95 * $G * $tp * $dc];
    # Plastic Stiffness
    set Kp [expr 0.95 * $G * $bf_c * ($tf_c * $tf_c) / $db];

    # Define Trilinear Equivalent Rotational Spring
    # Yield point for Trilinear Spring at gamma1_y
    set gamma1_y [expr $ Vy/$Ke];
    set M1y [expr $ gamma1_y * ($Ke * $db)];
    # Second Point for Trilinear Spring at 4 * gamma1_y
    set gamma2_y [expr 4.0 * $gamma1_y];
    set M2y [expr $M1y + ($Kp * $db) * ($gamma2_y - $gamma1_y)];
    # Third Point for Trilinear Spring at 100 * gamma1_y
    set gamma3_y [expr 100.0 * $gamma1_y];
    set M3y [expr $M2y + ($as * $Ke * $db) * ($gamma3_y - $gamma2_y)];

    # Hysteretic Material without pinching and damage (same mat ID as Ele ID)
    uniaxialMaterial Hysteretic $eleID $M1y $gamma1_y $M2y $gamma2_y $M3y $gamma3_y [expr -$M1y] [expr -$gamma1_y] [expr -$M2y] [expr -$gamma2_y] [expr -$M3y] [expr -$gamma3_y] 1 1 0.0 0.0 0.0

    element zeroLength $eleID $nodeR $nodeC -mat $eleID -dir 6
equalDOF $nodeR $nodeC 1 2
# Constrain the translational DOF with a multi-point constraint
# Left Top Corner of PZ
set nodeR_1 [expr $nodeR - 2];
set nodeR_2 [expr $nodeR_1 + 1];
# Right Bottom Corner of PZ
set nodeR_6 [expr $nodeR + 3];
set nodeR_7 [expr $nodeR_6 + 1];
# Left Bottom Corner of PZ
set nodeL_8 [expr $nodeR + 5];
set nodeL_9 [expr $nodeL_8 + 1];
# retained constrained DOF_1 DOF_2
#equalDOF $nodeR_1 $nodeR_2 1 2
#equalDOF $nodeR_6 $nodeR_7 1 2
#equalDOF $nodeL_8 $nodeL_9 1 2
}