UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Posgrados

Desarrollo de Modelos de Plasticidad Distribuida para Pórticos Especiales de Acero en el software Opensees

Proyecto de Investigación y Desarrollo

Edwin Andrés Zurita García

Ing. Pablo Andrés Torres Rodas, Phd
Director de Trabajo de Titulación

Trabajo de titulación de posgrado presentado como requisito
para la obtención del título de Magíster en Ingeniería Civil, Mención Diseño y Construcción de Estructuras Sismo Resistentes

Quito, abril 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE POSGRADOS

HOJA DE APROBACIÓN DE TRABAJO DE TITULACIÓN

Desarrollo de Modelos de Plasticidad Distribuida para Pórticos Especiales de Acero en el software Opensees

Edwin Andrés Zurita García

Nombre del Director del Programa:	Fernando Romo
Título académico:	Master of Science
Director del programa de:	Maestría en Ingeniería Civil

Nombre del Decano del colegio Académico:	Eduardo Alba
Título académico:	Doctor of Philosophy
Decano del Colegio:	Colegio de Ciencias e Ingenierías

Nombre del Decano del Colegio de Posgrados:	Hugo Burgos
Título académico:	Doctor of Philosophy

Quito, abril 2021

© DERECHOS DE AUTOR

Nombre del estudiante:	Edwin Andrés Zurita García
Código de estudiante:	00214303
C.I.:	0940843550
Lugar y fecha:	Quito, 30 de abril de 2021.

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following graduation project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

DEDICATORIA

Dedicado a mis padres, que siempre me han apoyado y ayudado en todos los aspectos, y les debo todos mis logros académicos y profesionales.

Para mis profesores, de los cuales aprendí siempre algo nuevo y me motivaron a seguir en la búsqueda de conocimiento.

Para mis familiares, que siempre han estado pendientes de mí y que de manera directa o incondicional me impulsaron a seguir adelante con mis proyectos.

AGRADECIMIENTOS

Agradezco a la Universidad San francisco de Quito por abrirme sus puertas y darme la oportunidad de demostrar mis aptitudes en su Maestría en Ingeniería Civil. A los profesores de la maestría cuyo conocimiento transmitido ayudó a la elaboración de este documento. Agradezco al profesor Pablo Andrés Torres Rodas, el cual siempre estuvo predispuesto en todo momento, para ayudarme en la elaboración del presente trabajo de titulación.

RESUMEN

El enfoque de este documento es principalmente comparar el comportamiento de diversas metodologías de modelación de edificios de acero con fibras, tomando en cuenta el deterioro por pandeo en las secciones, y los efectos P-Δ en los análisis estáticos no lineales. Por consiguiente, se estima que los desempeños serán los más cercanos a las realidades de los comportamientos de las estructuras. Por lo tanto, se decidió tomar estructuras de 2, 4, 8, 12 y 20 pisos propuestas por el NIST (National Institute of Standards and Technology), para así tener suficientes datos y poder verificar tendencias. Otros asuntos importantes de discusión fueron los materiales y el tipo de distribución a usar. Esto es fundamental en los modelos de fibras, puesto que el comportamiento de la estructura está directamente relacionado con la definición del material. Y el tipo de integración está relacionado con la resolución matemática del sistema. Como los softwares utilizados fueron Opensees y Matlab, Opensees da la oportunidad de definir materiales mediante varios tipos de modelos. Para este documento se enfocó en el uso de los materiales Steel02, Hysteretic y Parallel. Mientras que en las distribuciones se tomó como principales los tipos Gauss-Lobatto (GL) y Gauss Radau Modificado (GRM). Teniendo presente todos estos datos, se da una recomendación final de qué tipo de modelo de fibras, material y tipo de distribución usar, si se desea realizar un análisis similar.

Palabras clave: (Opensees, Matlab, Gauss-Lobatto, Gauss Radau Modificado, Steel02, Histeretico, material en paralelo, Fiber Hinge, Fibras, Rótula Plástica)

ABSTRACT

The focus of this paper is mainly to compare the behavior of various modeling methodologies using fibers, considering buckling deterioration in sections, and P-Δ effects in nonlinear static analyses. Consequently, it is estimated that the performances will be the closest to the realities of the structures' behaviors. Therefore, it was decided to take 2, 4, 8, 8, 12 and 20-story structures proposed by NIST (National Institute of Standards and Technology), to have enough data and to be able to verify trends. Other important issues of discussion were the materials and the type of distribution to be used. This is fundamental in fiber models, due to the behavior of the structure is directly related to the definition of the material. And the type of integration is related to the mathematical resolution of the system. As the software used were Opensees and Matlab, Opensees gives the opportunity to define materials by various types of models. For this paper we focused on the use of Steel02, Hysteretic and Parallel materials. While in the distributions, the Gauss-Lobatto (GL) and Modified Gauss Radau (GRM) types were taken as the main ones. Bearing in mind all these data, a final recommendation is given as to which type of fiber model, material, and type of distribution to use, if a similar analysis is desired.

Key words: (Opensees, Matlab, Gauss-Lobatto, Modified Gauss Radau, Steel02, Hysteretic, Parallel, Fiber Hinge, Fiber, Concentrated Plastic Hinge)

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPITULO 1.- INTRODUCCIÓN

Debido a la creciente innovación tecnológica en la ingeniería civil, es de conocimiento público la gran cantidad de softwares dedicados al análisis estructural. Sin embargo, se desconocen sus alcances o teorías detrás de su funcionamiento. Es necesario comprender que todo software en el mercado posee ventajas y desventajas al momento de realizar cálculos en el diseño estructural, y que ninguno puede capturar con eficacia todos los fenómenos físicos producidos en una estructura ante un evento sísmico. Dentro de este conjunto se encuentran las estructuras de acero, las cuales a comparación con las estructuras de concreto, también poseen grandes incertidumbres de comportamiento en el ámbito no lineal. Los softwares comerciales más reconocidos tienen la particularidad de simplificar el problema para así componer una interfaz más amigable con el usuario, debido a esto sacrifican precisión y realizan asunciones que la mayoría de las ocasiones el usuario pasa por alto. El ámbito no lineal de las estructuras es hasta el día de hoy uno de los campos con mayor diversidad de criterio entre los ingenieros y especialistas. Existen diversos tipos de enfoque, por lo que es necesario que el ingeniero tenga conocimientos previos de estos, antes realizar un análisis no lineal. Sumado a esto es necesario señalar que los pórticos SMF son los más vulnerables ante eventos sísmicos, debido a que su flexibilidad provoca un mayor daño en la estructura, en comparación con los pórticos arriostrados concéntricos. Pero su requerimiento arquitectónico suele ser muy solicitado.

Esta tesis está enfocada en proporcionar a los ingenieros estructurales una nueva herramienta capaz de agilizar el uso de un software estructural que no es muy reconocido en el ámbito profesional, y su uso se relega a aspectos investigativos debido a que posee un gran potencial de cálculo y análisis. El OpenSees es un software que permite a los usuarios modelar estructuras ya sea completas, o simplemente miembros estructurales de interés investigativo. Dándole la oportunidad a los ingenieros de realizar análisis de plasticidad concentrada, fibras

o elementos finitos, dependiendo de qué aspectos físicos se desee obtener información. Sin embargo, su interfaz con el usuario es poco amigable y depende mucho de que el ingeniero estructural sea adepto a lenguajes de programación. Por consiguiente, optimizar este proceso mediante un lenguaje de programación común entre los ingenieros civiles como MATLAB, da la oportunidad de que especialistas estructurales utilicen el software OpenSees de manera efectiva y rápida.

1.1. Opensees

Debido a que la ingeniería de terremotos siempre ha usado modelos y simulaciones para conocer el desempeño de las estructuras ante eventos sísmicos. Con ayuda de la "National Science Foundation" y el "Pacific Earthquake Engineering Research Center (PEER)", se desarrolló el "Open System for Earthquake Engineering Simulation" (Opensees), como software orientado a la investigación y simulación de sistemas estructurales y geotécnicos (University of Berkeley, 2006).

Por motivos concernientes a la arquitectura del programa (software libre), da la oportunidad que los usuarios realicen modelos más reales, agregando más datos sobre el comportamiento no lineal de las estructuras y materiales. De igual manera la modelación es flexible con el usuario, dándole la oportunidad de definir desde un espécimen de laboratorio hasta una estructura compleja. Otra ventaja muy remarcada es la amplia gama de algoritmos matemáticos iterativos de resolución y convergencia. Y la oportunidad de combinarlos a voluntad. Proveyendo así la capacidad de combinar diferentes tipos de elementos (University of Berkeley, 2006).

También es necesario acotar que Opensees posee una interfaz flexible en cuanto a adaptación computacional del Usuario. Puesto que adapta sus propiedades de almacenamiento, base de datos y "Network communication" (University of Berkeley, 2006).

1.2. Matlab

Es un software con un ambiente computacional numérico, destinado mayoritariamente para el análisis de datos, desarrollo de algoritmos y creación de modelos en general. (The Mathaworks, Inc, 1994-2021).

1.3. Modelos

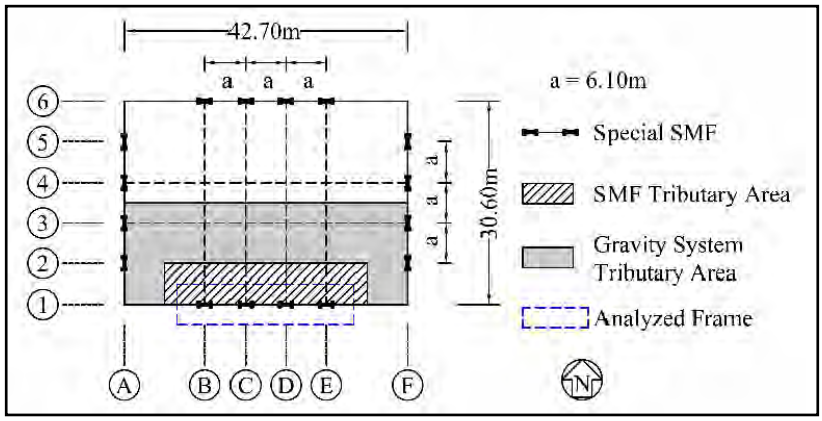
Los edificios propuestos provienen de los prototipos para pórticos resistentes a momento usados para la evaluación de la metodología FEMA P695 (NIST, 2010). Estos edificios tendrán las siguientes características:

1.3.1. Geometría.

Las alturas de los primeros pisos serán de 180 in y la de los entrepisos serán de 156 in. La longitud de vano será de 240 in, habiendo un total de 3 vanos en la estructura de análisis (ver Figura 1.1). Cabe recalcar que los empalmes de columna se los realiza en la mitad de la longitud de la columna. Y se utilizará la conexión RBS en todas las vigas de los modelos. Los paneles zonales a su vez constan de placas de refuerzo, cuyos espesores son previamente definidos en los prototipos (Tabla 1.-5.).

Figura 1.1

Planta tipo de edificios



Nota. Se observa el sistema lateral de análisis. Tomada de (NIST, 2010).

Tabla 1

Placas de refuerzo para paneles zonales en edificio de 2 pisos

Piso	Doubler Plate de columna interior (in)	Doubler Plate de columna exterior (in)
1	0	0
2	0	0

Nota. Fuente: National Institute of Standards and technology

Tabla 2

Placas de refuerzo para paneles zonales en edificio de 4 pisos

Piso	Doubler Plate de columna interior (in)	Doubler Plate de columna exterior (in)
1	0	5/16
2	0	5/16
3	0	5/16
4	0	5/16

Nota. Fuente: National Institute of Standards and technology

Tabla 3

Placas de refuerzo para paneles zonales en edificio de 8 pisos

Piso	Doubler Plate de columna interior (in)	Doubler Plate de columna exterior (in)
1	1/16	9/16
2	1/16	3/8
3	1/16	11/16
4	0	3/8
5	0	9/16
6	0	7/16
7	0	9/16
8	0	5/16

Nota. Fuente: National Institute of Standards and technology

Tabla 4

Placas de refuerzo para paneles zonales en edificio de 4 pisos

Piso	Doubler Plate de columna interior (in)	Doubler Plate de columna exterior (in)
1	0	1/2
2	0	7/16
3	1/16	5/8
4	1/16	5/8
5	0	5/8
6	0	5/8
7	1/16	11/16
8	1/16	11/16
9	0	9/16
10	0	9/16
11	1/16	9/16
12	1/16	9/16

Nota. Fuente: National Institute of Standards and technology

Tabla 5

Placas de refuerzo para paneles zonales en edificio de 8 pisos

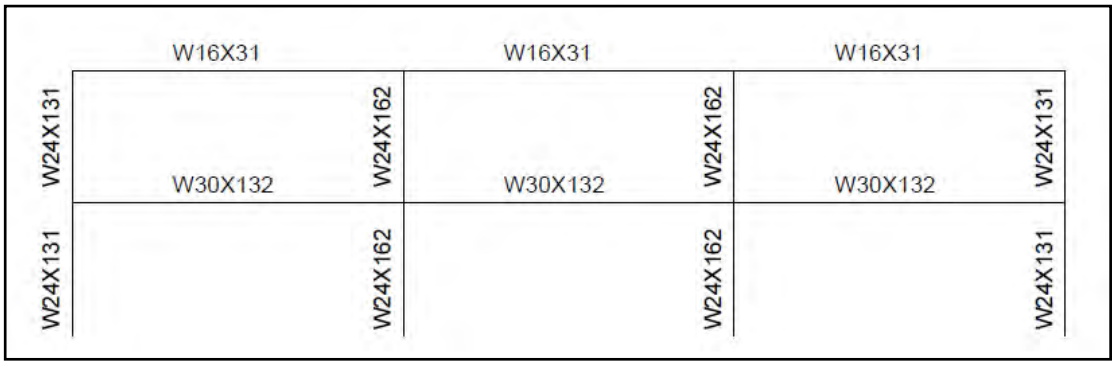
Piso	Doubler Plate de columna interior (in)	Doubler Plate de columna exterior (in)
1	0	1/4
2	0	1/4
3	0	1/4
4	0	1/4
5	0	1/4
6	0	1/4
7	0	1/4

Piso	Doubler Plate de columna interior (in)	Doubler Plate de columna exterior (in)
8	0	1/4
9	0	5/16
10	0	5/16
11	0	1/2
12	0	1/2
13	1/16	1/2
14	1/16	1/2
15	1/4	9/16
16	1/4	9/16
17	3/8	9/16
18	3/8	9/16
19	0	3/16
20	0	3/16

Nota. Fuente: National Institute of Standards and technology

Figura 1.2

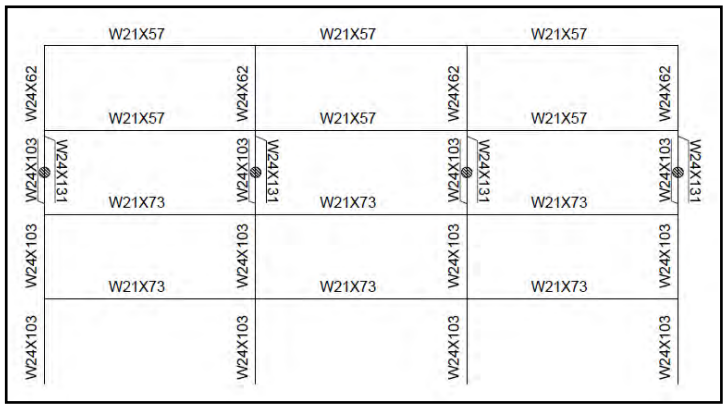
Secciones usadas en edificio de 2 pisos



Nota. Adaptado de (NIST, 2010).

Figura 1.3

Secciones usadas en edificio de 4 pisos

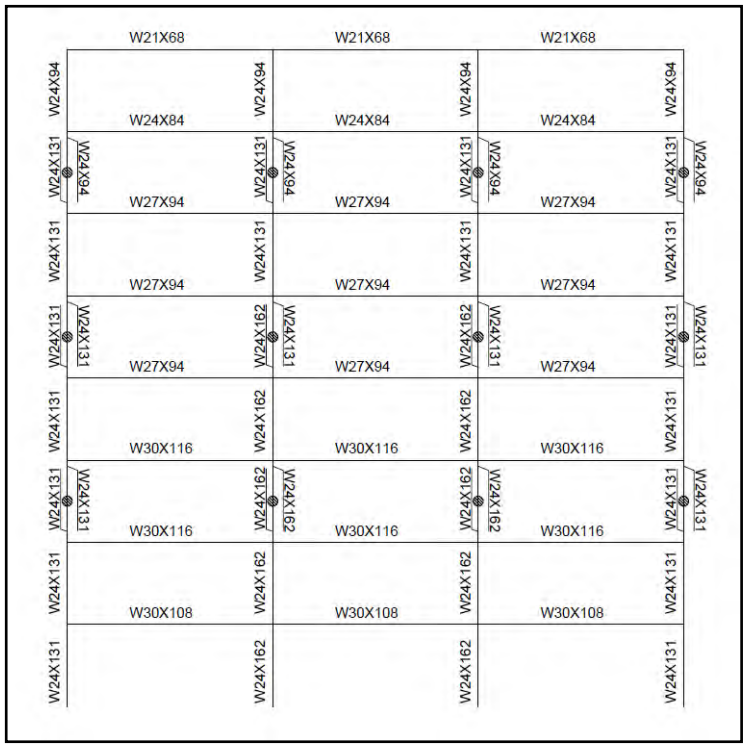


Nota. En el grafico también se muestran las ubicaciones de los empalmes de columna.

Adaptado de (NIST, 2010).

Figura 1.4

Secciones usadas en edificio de 8 pisos

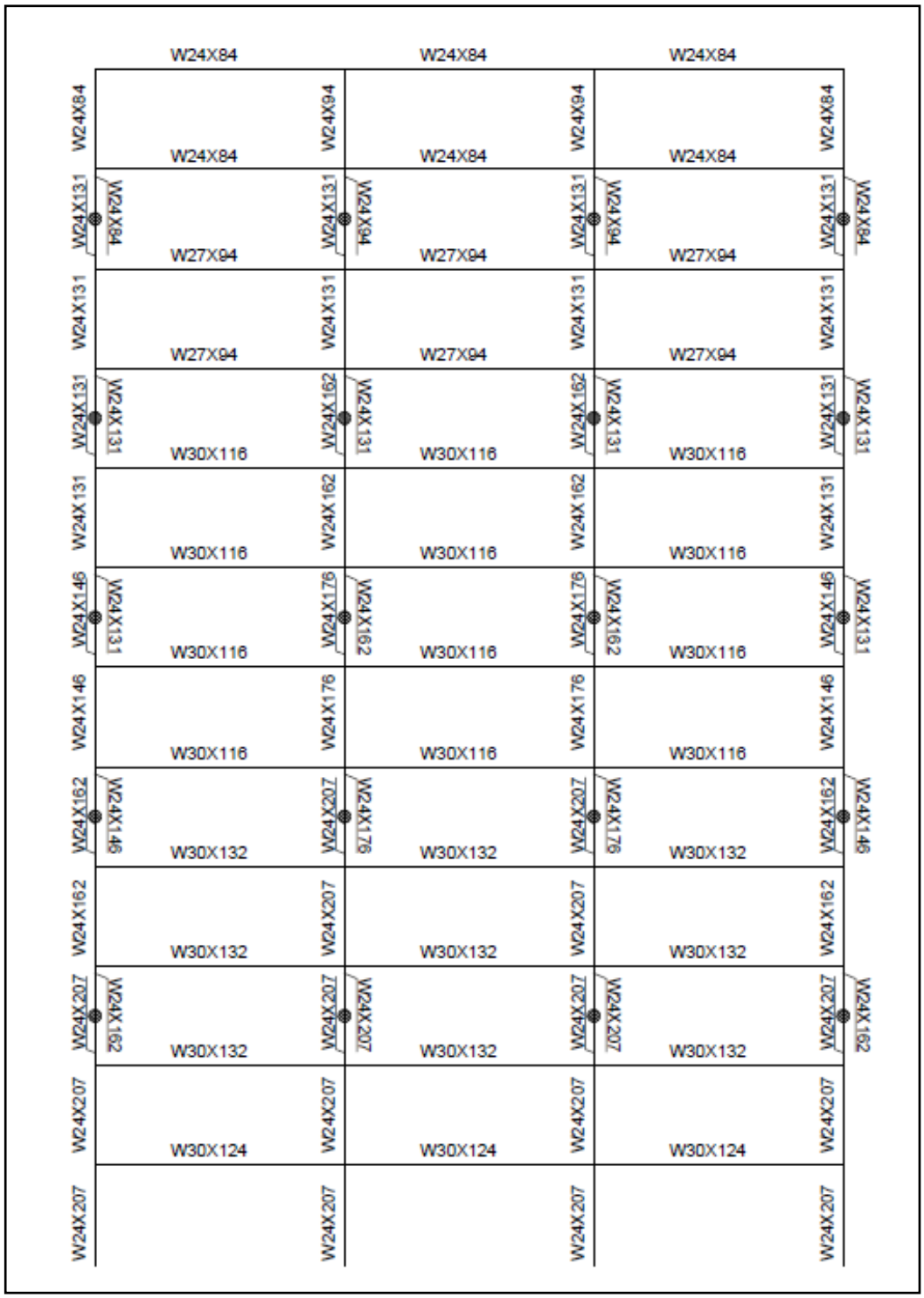


Nota. En el grafico también se muestran las ubicaciones de los empalmes de columna.

Adaptado de (NIST, 2010).

Figura 1.5

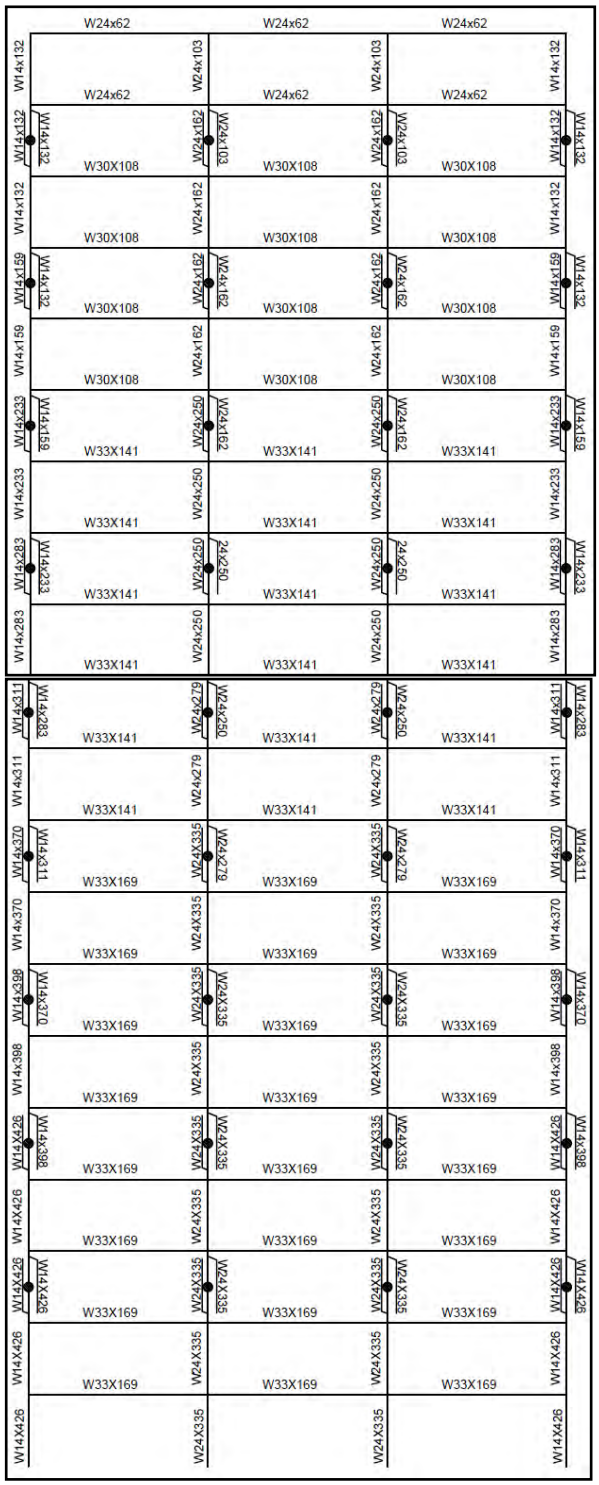
Secciones usadas en edificio de 12 pisos



Nota. En el grafico también se muestran las ubicaciones de los empalmes de columna. Adaptado de (NIST, 2010).

Figura 1.6

Secciones usadas en edificio de 12 pisos



Nota. En el grafico también se muestran las ubicaciones de los empalmes de columna. Adaptado de (NIST, 2010).

Como se puede observar, los modelos no son tan complejos como podrían serlo. Esto debido a que se pasa por alto ciertas condiciones reales, que en cierta medida podrían afectar el comportamiento de la estructura. Algunas de estas podrían ser el desgarre o corte de los pernos en las conexiones o el pandeo placas por compresión. Adicionalmente también se obvia el efecto del sistema de piso en la estructura, el cual dependiendo de su ubicación y diseño puede suponer una condición desfavorable para las columnas (NIST, 2017). Otro punto para resaltar es que la longitud de la rótula plástica para el método de fibras sigue siendo un tema de debate e investigación, hasta la actualidad. En este proyecto solo usaron 2 longitudes, las cuales están debidamente referenciadas y calibradas.

1.3.2. Masas y Pesos.

Se utilizará un peso de primer piso igual a 800.45 kips, para pisos intermedios será de 796.70 kips y para el techo se usará 720.38 kips (NIST, 2010). Las masas reactivas serán asignadas en un nodo determinado del panel zonal, cuyos valores se describen en la tabla 6.

Tabla 6

Masas reactivas en las estructuras

Piso	Masa en C1 (kip*s²/in)	Masa en C2 (kip*s²/in)	Masa en C3 (kip*s²/in)	Masa en C4 (kip*s²/in)
1	0.62147	0.41431	0.41431	0.62147
Intermedios	0.61856	0.41237	0.41237	0.61856
Techo	0.5593	0.37287	0.37287	0.5593

Nota. Fuente: National Institute of Standards and technology

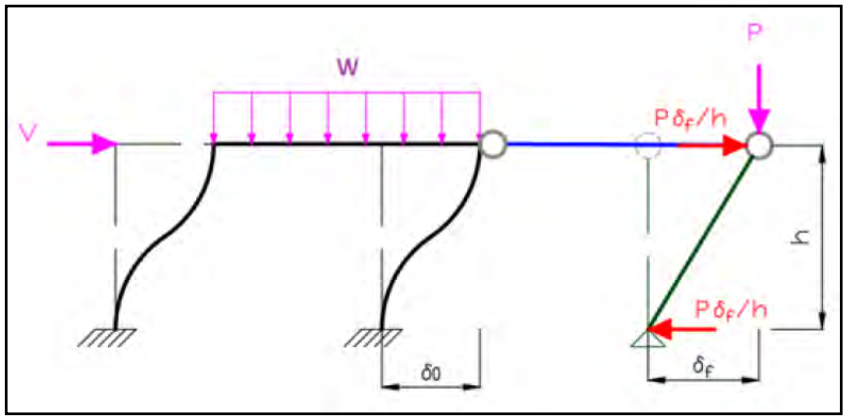
1.3.3. Leaning Column.

Los efectos de segundo Orden (P-Delta) hacen referencia a una reducción de rigidez debido a la acción de cargas gravitacionales en el equilibrio de la estructura deformada (Kassimali, 2010). El FEMA P-695 propone introducirlo dentro del análisis no lineal mediante el método de columna fantasma (Leaning Column). En el cual se agrega una columna adicional

sin rigidez lateral al modelo porticado. Para de esta manera capturar la contribución de la carga no tributaria al pórtico (ver Figura 1.7).

Figura 1.7

Esquema del comportamiento de la Leaning Column



Nota. Tomado de (Astudillo Carpio, 2018).

1.3.4. Paneles Zonales.

Desde el punto de vista del modelo, estos elementos constan de 4 elementos rígidos unidos en las esquinas mediante resortes (Astudillo Carpio, 2018). Cuyo comportamiento está definido por la fluencia del alma de la columna, reacomodo y fluencia en los patines de la columna, y finalmente endurecimiento por deformación. Para simular este comportamiento, los resortes deben ser definidos mediante una curva trilineal.

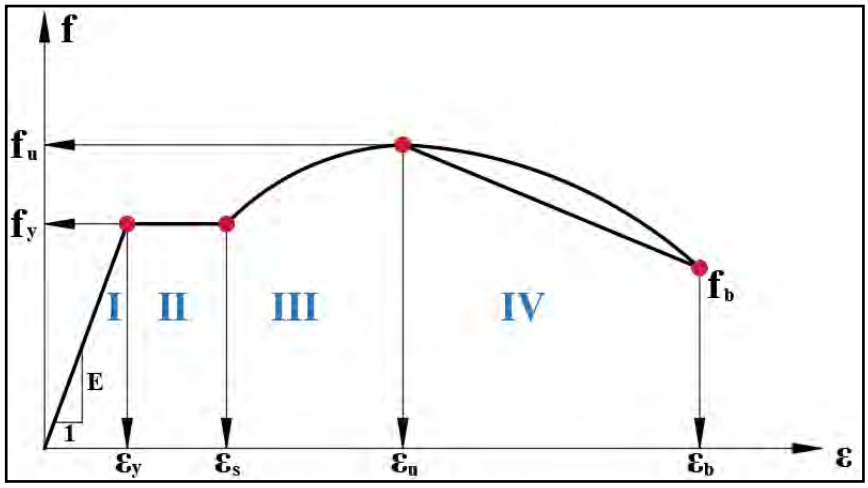
1.4. Materiales

El material para usarse en las estructuras es el ATM A992, el cual posee un módulo de elasticidad (E) de 29000 Ksi, un esfuerzo de fluencia (Fy) de 50 Ksi, la deformación de fluencia (Ɛy) es de 0.17%, con un coeficiente de variación de fluencia (ry) de 1.1 y una deformación de plasticidad (Ɛs) del 2.4% (ASTM, 2004). El correspondiente esfuerzo último (Fu) será de 65 Ksi, con deformación última (Ɛu) de 21%, de igual manera un coeficiente de variación último

(rt) de 1.1, finalmente el esfuerzo de ruptura (Fb) es de 30 Ksi y su deformación de ruptura es de 54% (ASTM, 2004). (ver Figura 1.8)

Figura 1.8

Curva característica del acero



Nota. Tomado de (Lee, Engelhardt, & Jeong Choi, 2015).

Para simular este comportamiento, el Opensees ofrece varias alternativas, como es el material steel02, material Hysteretic, o incluso la posibilidad de usar 2 tipos de materiales en paraleo (Parallel).

1.4.1. Steel02.

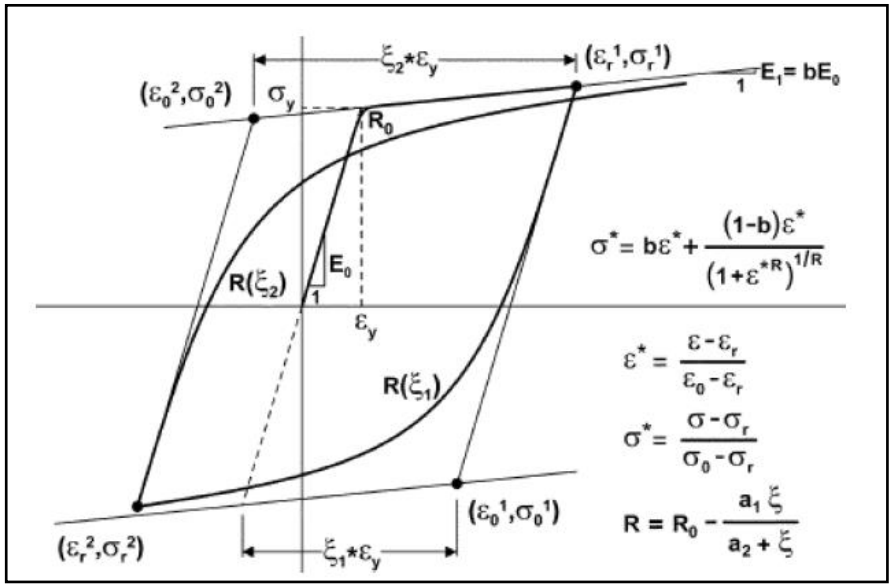
El material steel02 toma como referencia la propuesta hecha por Giuffreé-Menegotto-Pinto (ver Figura 1.9) y se lo usa para la modelación cíclica del acero estructural (Martinez Pesantez & Pozo Ocampo, 2018). Este material consta de 3 partes fundamentales, la primera de una rigidez inicial, seguido por una de transición exponencial, llegando hasta un endurecimiento del material (Bosco, Ferrara, Ghersi, Marino, & Rossi, 2014). Los datos necesarios para ingresar el material en Opensees son Fy, E, b, R0, cR1 y cR2, cuyos valores serán 50, 29000, 0.0026, 20, 0.925 y 0.15 respectivamente. Adicionalmente también se necesita parámetros a1, a2, a3 y a4, los cuales serán -0.01, 2.1, -0.01, 2.1, calibrados mediante el

experimento de viga en voladizo (Martinez Pesantez & Pozo Ocampo, 2018). Este material, generalmente no tiene problemas de convergencia en la parte numérica, sin embargo es complicado incluir el deterioro debido los pandeos en la sección.

Sintaxis: “uniaxialMaterial Steel02 $Tag $fy $E $b $R0 $cR1 $cR2 $a1 $a2 $a3 $a4”

Figura 1.9

Formulación del material steel02



Nota. Tomado de (Orakcal & Wallace, 2006).

1.4.2. Hysteretic.

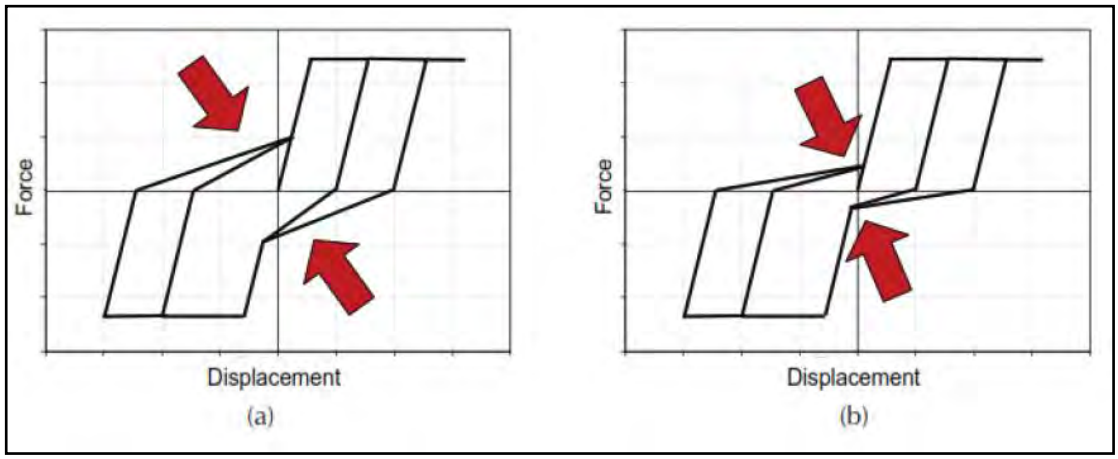
El material Hysteretic le da la facilidad al usuario de construir una curva trilineal con reducción de resistencia por ductilidad y por energía, además de degradación de rigidez basada en ductilidad (Martinez Pesantez & Pozo Ocampo, 2018). Para el ingreso del ASTM A992 es necesario definir los parámetros de Fy, Ɛy, Fu, Ɛu, Fb y Ɛb a manera de coordenadas. Es decir (Fy, Ɛy), (Fu, Ɛu) y (Fb, Ɛb). Adicionalmente se deben definir los valores referentes al pinching, los cuales se refieren a la reducción de rigidez de recarga junto con la recuperación de rigidez, cuando surge el desplazamiento en la dirección opuesta (Astudillo Carpio, 2018) (ver Figura

1.10). Para este proyecto se usarán valores de 0.50 para el sentido “X” y 0.75 para el sentido “Y”, además de usar un Fu de 71.5 y no de 65 por motivos de convergencia. Tampoco se considerará daño por resistencia o por energía según el modelo propuesto por Pozo et al., 2018, por lo que los valores dam 1 y dam 2 serán de 0. Este tipo de material es el que presenta mayores problemas de convergencia (Martinez Pesantez & Pozo Ocampo, 2018).

Syntaxis: “uniaxialMaterial Hysteretic $Tag $fy $ey $fu $eu $fb $eb -$fy -$ey -$fu -$eu -$fb -$eb $pinchx $pinchy $dam1 $dam2”

Figura 1.10

Comportamiento Pinching



Nota. a) Comportamiento moderado b) Comportamiento severo. Tomado de (FEMA, 2009).

1.4.3. Parallel.

El Parallel da la oportunidad de ejecutar dos materiales previamente definidos, esto lo realiza manteniendo las deformaciones , pero sumando las rigideces y los esfuerzos (Martinez Pesantez & Pozo Ocampo, 2018). Esto no se debe tomar como un material, si no como una herramienta que crea uno. En este proyecto se empleará este método utilizando los dos anteriores para formar una constitutiva que represente un comportamiento no simétrico, para

que se capture el pandeo local representativo de las secciones W. (Pozo S. , Astudillo, Samaniego, & Flores, 2020)

Syntaxis: “uniaxialMaterial Parallel $Tag $Tag1 $Tag2 -factors 1 -1”

1.4.4. Steel01.

Es el modelo más básico, compuesto por una curva bilineal que representa el comportamiento del acero. Esta constitutiva no requiere datos de deterioro, solo fluencia, modulo de elasticidad y parámetros de hardening. Este material se lo usara únicamente para definir los resortes de los paneles zonales, en conjunto con el comando parallel, para así dar origen a un nuevo material que tome en cuenta las placas de refuerzo.

Syntaxis: “uniaxialMaterial Steel01 $matTag $Fy $E0 $b”

1.5. Análisis estático no lineal

El procedimiento se basa en aplicar un análisis estático monótónico, conocido comunmente como “Pushover” (Pozo S. , Astudillo, Samaniego, & Flores, 2020). Practicamente lo que se hace es empujar la estructura en base a un patrón de cargas determinado, hasta alcanzar un desplazamiento tal que permita capturar la capacidad completa de la estructura. El análisis se encuentra calibrado en base a los parámetros propuestos por el FEMA P-695. En el cual indica establecer el patrón de cargas en base al primer modo de vibración (Pozo S. , Astudillo, Samaniego, & Flores, 2021 en revisión). En este proyecto los resultados serán presentados como, la relacion entre el cortante basal normalizado con el peso sísmico de la estructura y la deriva máxima de techo (capacidad de la estructura).

1.6. Análisis Dinámico

Un anális dinámico tiempo-historia involucra someter a la estructura a un set de registros debidamente escalados hacia una intensidad solicitada. Y luego evaluar bajo criterios específicos la mediana de las máximas respuestas. Sin embargo, para el objetivo de este

documento (revisar el comportamiento de los diagramas de momento-curvatura en las fibras de los diferentes modelos) solo se utilizará un registro. El cual fue seleccionado debido a su duración e intensidad. El registro es el correspondiente al evento sísmico producido en la ciudad de Chichi en Taiwan en el año de 1991.

1.7. Diagrama Momento-Curvatura

El diagrama momento-curvatura da la oportunidad de conocer el potencial de incursión en el rango no lineal de una sección antes de llegar al fallo. Dado que la curvatura es M/EI, bastaría con conocer el módulo de elasticidad y la Inercia para construir la curva. Siempre y cuando la falla sea dominada por flexión. (Mora & Aguiar, 2015). Estos datos se los obtendrá solamente de las fibras inferiores de las columnas del primer piso.

1.8. Análisis Numérico

Las bases de la resolución numérica se asientan en el comportamiento de una estructura de 1 dof (degree of freedom). En donde se busca resolver la ecuación fundamental del movimiento, en la cual la fuerza externa puede ser producto de una aceleración de piso (análisis dinámico) o producto de un desplazamiento (análisis pushover). Dicha ecuación se puede resolver mediante “diferencia central” (método explicito) o “Newmark” (método implícito), por lo que este último se resuelve de manera iterativa. Sin embargo, el método de diferencia central se vuelve iterativo cuando se incluye aspectos de no linealidad. Mientras que el método de Newmark necesita hacer uso del método de Newton-Raphson para incluir este aspecto en su formulación. Debido a que el método de Newton-Raphson hace uso de la tangente en un punto de la función, es necesario generalizar esa tangente mediante la aproximación de Taylor. Este proceso el método de Newmark lo controla en base al “Residuo” (tolerancia) o diferencia entre el objetivo y la iteración. Finalmente, si se expanden estos métodos a varios grados de libertad,

se necesita desacoplar el sistema de ecuaciones y resolver cada sistema de manera independiente y simultánea. Esto último es básicamente lo que realiza Opensees en los modelos.

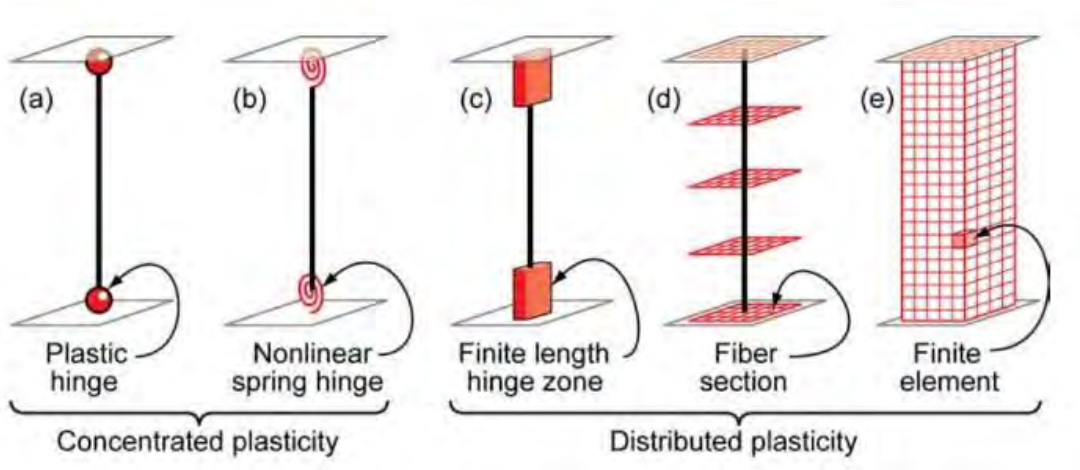
CAPITULO 2.- MODELACIÓN CON FIBRAS

Dentro del ambito de la modelación de estructuras existen 4 tipos, “CPH” (Concentrated Plastic Hinge), “fiber Hinge”, “distributed plasticity” y “finite element” (ver Figura 2.1). En este proyecto se indagará en los aspectos de fiber Hinge y distributed plasticty. Sin embargo, ciertos elementos utilizarán CPH tales como los paneles zonales y las bases. Además de exponer tambien un modelo en el cual las vigas sean definidas como CPH y las columnas como “fiber hinge”, debido a que este modelo tambien se encuentra referenciado y se desea observar como afectará a las estructuras.

El método de manera resumida plantea la discretización de la sección en puntos específicos del elemento, a estos puntos se los denomina puntos de control de la integración numérica (Filippou, Spacone, & Taucer, 1996) (ver Figura 2.2).

Figura 2.1

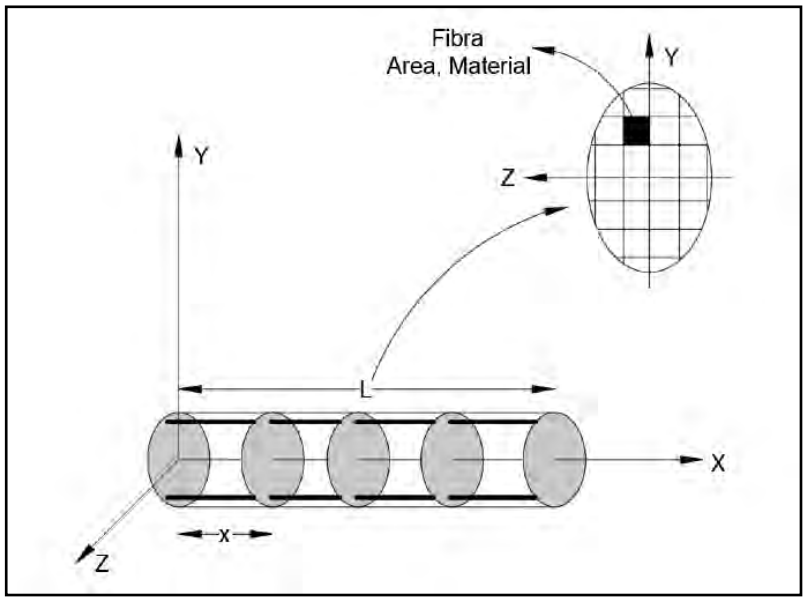
Modelos idealizados de elementos viga-columna



Nota. Tomado de (NIST, 2010).

Figura 2.2

Discretización de la sección y puntos de control



Nota. En la imagen se logra apreciar una viga con 5 puntos de integración, y la discretización de una sección circular, a manera de ejemplo. Tomado de (Filippou, Spacone, & Taucer, 1996).

La formulación del elemento está regida por la asunción de distintas hipótesis cinemáticas. Una de ellas es la de linealidad geométrica (Euler-Bernoulli). “En la cual las secciones planas permanecen planas y perpendiculares al eje longitudinal durante toda la historia de deformación del elemento” (Filippou, Spacone, & Taucer, 1996). Por lo tanto, este comportamiento hace que las deformaciones axiales de la sección transversal sean asociadas solamente a las deformaciones axiales del centroide, en combinación con la curvatura alrededor de los ejes mayor y menor de la sección. Luego la deformación axial de cada fibra es asociada al esfuerzo axial de cada fibra de manera correspondiente. Y finalmente se realiza la integración de resultados de cada una de las fibras en la sección, obteniendo así los esfuerzos, fuerzas y momentos respectivos de la sección (Martinez Pesantez & Pozo Ocampo, 2018). Sin embargo,

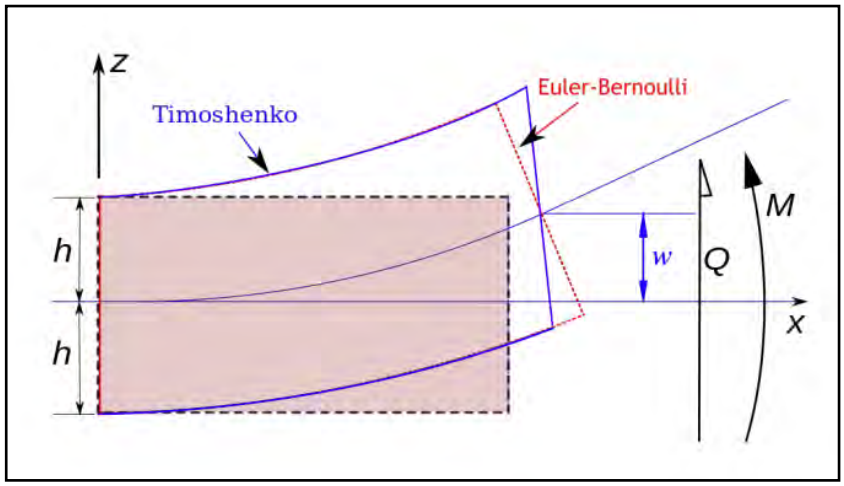
este comportamiento limita el método de fibras a la captura de solo un tipo de fenómeno. Otra de las hipótesis es la de "Teoría de Vigas de Timoshenko" (ver Figura 2.3), en la cual incluye en la derivación la deformación por corte. Debido a esto es posible incluir en el método de fibras una combinación de efectos fenomenológicos, como pueden ser, el caso más común de fuerza axial y momento (P-δ o M-ϕ). Aunque también da la oportunidad de hacer una combinación entre corte y torsión (V-γ o T-ϕ_x). Sin embargo, estas ecuaciones son poco estandarizadas, por lo que su estudio podría ser tema para futuras investigaciones (NIST, 2017).

La teoría de fibras también está dividida en dos tipos de enfoque. Las fibras controladas por fuerzas y las fibras controladas por deformación. Aunque en ambos se comparte la capacidad de formar rótulas plásticas en cualquier posición del elemento. Las fibras controladas por desplazamiento son un método aproximado, mientras que las fibras controladas por fuerzas si tratan de satisfacer (de manera exacta) las condiciones de equilibrio en la sección, inclusive en el rango no lineal (Martinez Pesantez & Pozo Ocampo, 2018). Esto ocurre porque dentro de la formulación, las "controladas por deformaciones" obtienen las respuestas de desplazamiento directamente de la matriz de desplazamiento del elemento, mediante funciones de interpolación. Y luego determinan las fuerzas resistentes y la rigidez de la sección de la curva esfuerzo deformación. Por lo tanto, para solucionar la confiabilidad de estos resultados se necesitaría hacer un mallado mucho más fino. En contraste las "controladas por fuerzas" asumen que la distribución del momento flector es lineal (siempre y cuando no haya carga distribuida en el elemento), el cortante es constante y la carga axial es constante, todo esto a lo largo del elemento. Esto hace que el equilibrio estático entre las fuerzas externas e internas sea exacto, independientemente de la no linealidad del material. Una vez realizado el equilibrio se determinan las fuerzas en la sección, mediante funciones exactas de interpolación. Por consiguiente, los únicos errores vendrían de la convergencia de la parte numérica, la cual se

puede mejorar incluyendo mayor número de puntos de integración (Martinez Pesantez & Pozo Ocampo, 2018).

Figura 2.3

Comparativa entre Euler-Bernoulli y Timoshenko



Nota. M es el momento flector, Q es la fuerza cortante, w el desplazamiento del eje y 2h la altura de la viga. Tomado de (Nguyen, 2017).

Aunque los modelos con fibras tienen la capacidad de tomar en cuenta varios aspectos fenomenológicos, siguen teniendo el inconveniente de que no siempre son capaces de predecir límite de ductilidad y por ende las pérdidas de capacidad (Powell, 2010). También es necesario reconocer el hecho de que según Powell ciertas veces el esfuerzo cortante en columnas de acero suele controlar el comportamiento del elemento. Si bien es cierto que el modelo de fibras da la oportunidad de combinar efectos como carga axial y momento. Hacer una combinación entre carga axial, momento y esfuerzo cortante (lo cual es lo necesario de hacer si se quiere incluir los efectos de esfuerzos cortantes) requiere de un modelo mucho más complejo, en comparación con el modelo de fibras tradicional. Otra limitación importante es que al trabajar mediante el supuesto de que deformaciones planas permanecen planas, las deformaciones del elemento no

pueden ser muy grandes, lo cual podría no cumplirse en los rangos no lineales (Martinez Pesantez & Pozo Ocampo, 2018).

Es necesario comprender también la importancia de los putos de integración, así como su adecuada cantidad y distribución, para tratar de afinar los modelos. Dentro de las variantes de distribuciones que presenta el Opensees. Este documento se enfocará en 2, Gauss-Lobatto y Gauss-Radau modificado, debido a que son las usadas en las referencias. En la distribución Gauss-Lobatto, se prioriza la captura de esfuerzos en los extremos de los elementos, debido a que se asume que no hay cargas intermedias en el elemento (Scott, 2011). Por consiguiente, se requieren un mínimo de 3 puntos de integración para que se pueda solucionar el equilibrio de fuerzas en el elemento. Mientras que en la distribución Gauss-Radau modificada, se deben especificar detalladamente las longitudes de rótulas plásticas inicial y final del elemento. En este método de integración no se da información de los puntos de integración, ya que le método los genera automáticamente (Pozo S. , Astudillo, Samaniego, & Flores, 2021 en revisión). Como se podrá intuir, este método de integración es el más adecuado para los modelos de "fiber hinge". Sin embargo, se debe tener cuidado con las longitudes de rótula plástica, puesto que valores muy grandes podrían ocasionar una superposición de puntos de integración y obtener resultados errados. Una recomendación sería que la longitud de rótula plástica se limite al 20% de la longitud del elemento para evitar la superposición (Pozo S. , Astudillo, Samaniego, & Flores, 2021 en revisión).

Teniendo en cuenta todos estos aspectos teóricos, se podrá mediante la metodología a continuación, analizar como varía el comportamiento de las estructuras propuestas ante diversos tipos de constitutivas de materiales y modelaciones.

CAPITULO 3.- REVISIÓN DE LITERATURA

En este proyecto se utilizó como introducción a la modelación con fibras el documento de Guidelines for Nonlinear Structural Analysis for Design of Buildings Part IIa – Steel Moment Frames propuesto por el NIST. Sin embargo, el documento no contenía aplicativos hacia la modelación aplicada con Opensees. Por lo tanto, el tutor de tesis sugirió revisar la fuente de "Análisis por desempeño de una estructura especial de acero modelada con fibras controladas por fuerzas en el software OpenSees" con autoría de Sebastián Pozo y Lisseth Martínez. En donde se encontró aparte de una mayor cantidad de referencia que revisar, para sustentar la parte teórica. También se proponía una modelación de estudio aplicativa con Opensees. A partir de aquí se trató de investigar nuevas fuentes que promovieran modelos diferentes, por motivos comparativos. En consecuencia, se obtuvieron los documentos "Regularization Method to Include Material Softening In Fiber Beam Column Elements For Seismic Performance Assesment Of Steel Frames" y "Objective Phenomenological Constitutive Law for Collapse Analyses in Distributed Plasticity Steel Frame Models", ambos con autoría de Sebastián Pozo, Bryam Astudillo, Esteban Samaniego y Francisco Flores. Estas últimas referencias contenían no solo nuevos tipos de modelos, si no que también proveyeron de tolerancias en ciertos parámetros, las cuales se las tomó en cuenta para los nuevos modelos. Finalmente se revisó casi todas las referencias asociadas a estos documentos, para así comprender los alcances de las modelaciones y sus limitaciones.

CAPITULO 4.- METODOLOGÍA Y DISEÑO DE LA INVESTIGACIÓN

Para el desarrollo y análisis de los modelos se utilizarán dos tipos de softwares, Matlab y Opensees. El Matlab generará los archivos necesarios para que el Opensees los ejecute posteriormente.

Los archivos que ejecuta el software Opensees son de extensión TCL (tool command language). Por consiguiente, es necesario que todos los archivos obedezcan este lenguaje de programación.

El primer modelo que se realizará será el edificio base de 8 pisos, luego de calibrar este ejemplar se procederá a configurar los correspondientes archivos para edificios de 2, 4, 12 y 20 pisos. Sin embargo, modelar un edificio de 8 pisos involucrando todos los componentes estructurales necesarios para que la respuesta sea la más representativa de la estructura, en lenguaje TCL, se vuelve demasiado laborioso. Por lo tanto, se utilizará como ayuda el software Matlab, el cual mediante bucles y la función “fprintf” es capaz de generar las líneas de datos necesarias en formato TCL. Y luego simplemente modificando el número de pisos se generarán los archivos necesarios para los edificios de 2, 4, 12 y 20 pisos.

4.1. Modelación

Este apartado se enfocará en el modelo de 8 pisos, ya que para los demás modelos el procedimiento será el mismo, simplemente variará el hecho de que los archivos tengan más o menos líneas debido al número de pisos.

Para que el Opensees construya un modelo es necesario brindarle la siguiente información: Geometría, Elementos, Materiales, Constraints, Resortes, Masas y para este proyecto, división de fibras en las secciones. La modelación de las fibras dependerá de las propuestas revisadas por diferentes autores. Para este estudio se utilizarán dos propuestas de modelación calibradas por sus respectivos autores.

En ambas se coincide con la modelación de la conexión RBS (Reduced Beam Section), pero se difiere en la modelación de las fibras de las columnas. La primera propuesta requiere una longitud de rótula plástica de L/12(donde L es la altura de entrepiso), además de subdividirla en tres elementos los cuales deberán poseer al menos tres puntos de integración (Martinez Pesantez & Pozo Ocampo, 2018). La segunda propuesta establece usar una longitud de rótula plástica igual a la altura de la sección de la columna y con al menos 5 puntos de integración. (Pozo S. , Astudillo, Samaniego, & Flores, 2021 en revisión)

Ya una vez establecido que modelo de fibras se usará, se procede diagramar la geometría de la estructura mediante nodos. Los nodos deben ser tales que abarquen todos los componentes estructurales, como vigas, columnas, zona de paneles y empalmes.

El Opensees lee los nodos de la estructura de la siguiente manera: “node Tag coordenada X coordenada Y” en ese orden. El tag (etiqueta) es vital en el Opensees, ya que mediante este el programa reconoce que tipo de elemento es, ya sea nodo, viga, columna, etc. Por eso es importante llevar un control de los tags utilizados.

En el caso de los nodos de las columnas se utilizará la siguiente nomenclatura del tag: “xya”, donde “X” representa la ubicación del pilar y puede ser de 1 a 5 tomando en cuenta la leaning column. Y “Y” el nivel del pilar y su límite está ligado al número de pisos. En este modelo irá de 1 a 9 tomando en cuenta 1 como la base de la estructura. El valor de “A” se utilizará para indicar la posición del nodo con respecto al panel zonal, 5 indicando abajo del panel zonal y 7 arriba del panel zonal. En la base (nivel 1) se usará solo el valor de 7 (ejemplo 117), aparte se definirán otros nodos solo con coordenadas en x (ejemplo 11), ya que se los definirá como empotramientos (cimentación). Y para modelar la unión de la cimentación con la superestructura se definirán resortes que representen este vínculo. Para los empalmes (splices se usará de igual manera “xya”, solo que “A” tomará valores de 91 y 92. Aunque tengan las

mismas coordenadas 91 se lo usará para unir la parte superior del panel zonal con el splice, y 92 para unir la parte inferior del panel zonal con el splice. Posteriormente se usará un constraint entre 91 y 92 para darle continuidad a las columnas. Hasta este punto se tienen definido los nodos de los extremos de las columnas y los splices en los pisos correspondientes, pero cabe recalcar que el tipo de análisis será en base a fibras. En este modelo se usará el método de "fiber hinge" el cual solo analiza mediante fibras una longitud determinada (longitud de rótula plástica), y no toda la longitud del elemento (NIST, 2017). El mismo concepto se usará para la modelación de vigas. Por lo tanto, es necesario también definir los nodos de los elementos fibra, los cuales para las columnas tendrán la siguiente etiqueta: "xy0za". Como la longitud de rótula plástica es L/12 y está subdividida en 3 elementos, "X" y "Y" mantienen su significado, "Z" es la posición de la fibra y los valores van de 1 a 3, mientras que "A" es la convención para la ubicación de las fibras en las columnas siendo 1 en la parte baja de la columna y 2 en la parte superior de la columna.

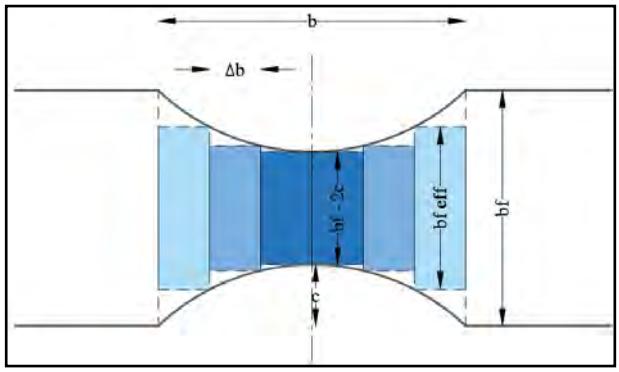
Para los paneles zonales se usarán tags con notación "xybc". Donde "X" y "Y" tienen el mismo significado que en las columnas, pero "BC" hace referencia a la ubicación relativa de la unión viga-columna. Siendo las convenciones 001 y 002 para arriba izquierda, 003 y 004 para arriba derecha, 006 y 007 abajo derecha, 008 y 009 abajo izquierda, 005 en medio derecha y 100 en medio izquierda.

En las vigas ya se tiene definido los nodos extremos (proveniente de las partes medias de los paneles zonales). Por lo tanto, solo faltaría definir los nodos de las fibras. El tag es prácticamente igual al de las columnas (xy0za), solo que el valor de z toma valores de 1 a 7, ya que la subdivisión de la zona de rótula plástica (zona reducida de la conexión RBS) se hará de 6 elementos. Y los valores de "A" son de 4 para izquierda y 3 para derecha. Para ampliar un

poco más la explicación se mostrarán, la discretización de la zona reducida de la conexión RBS (ver Figura 4.1), y los esquemas de la modelación (ver figura 4.2 a 4.4)

Figura 4.1

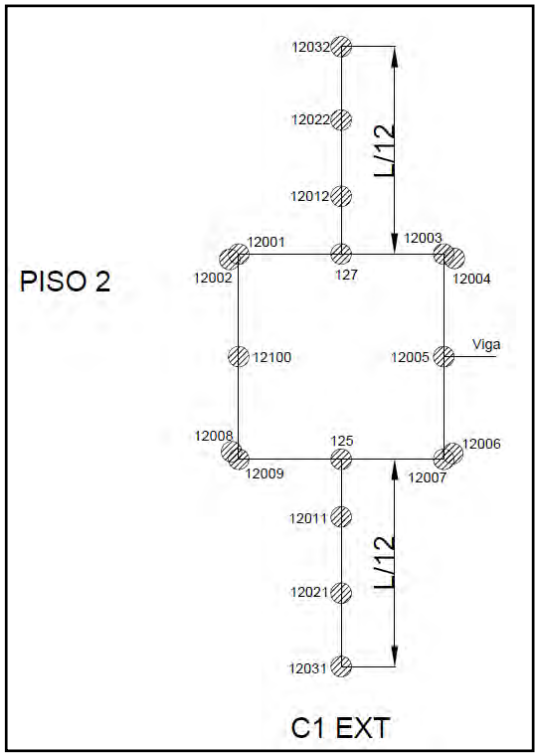
Discretización del elemento RBS



Nota. En total serán 6 elementos (Martinez Pesantez & Pozo Ocampo, 2018).

Figura 4.2

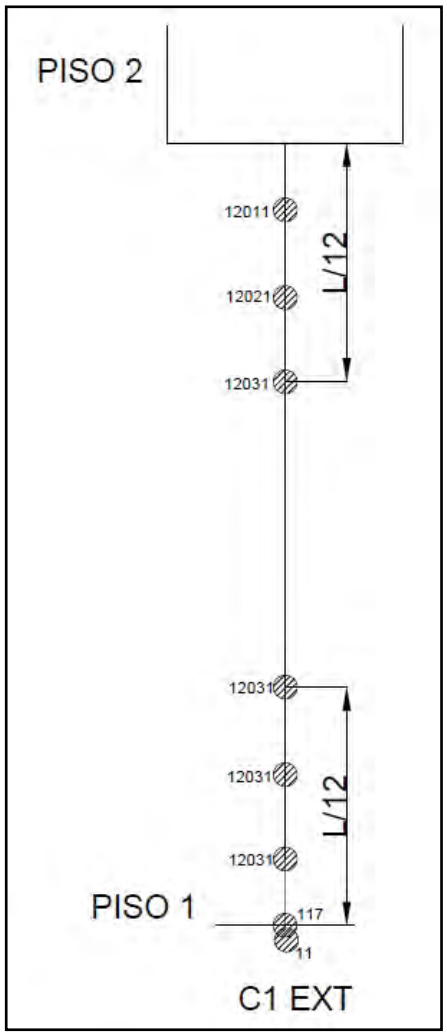
Esquema de nodos en panel zonal



Nota. Para la modelación se toma como piso 1 la base

Figura 4.3

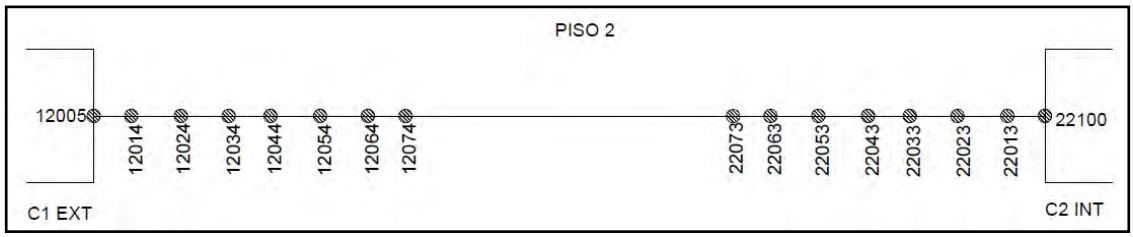
Esquema de nodos en columnas



Nota. Los nodos traslapados indican las mismas coordenadas.

Figura 4.4

Esquema de nodos en vigas



Nota. Este esquema se repite en todas las vigas.

En este modelo se usarán dos tipos elementos principalmente, los elementos para miembros elásticos (element elasticBeamColumn) y los elementos de fibras controladas por fuerzas (element forceBeamColumn). También es necesario definir los elementos del leaning columna, en los cuales los pisos estarán unidos mediante elementos tipo barra (element truss), mientras las columnas con elementos elásticos.

Los comandos para ingresar estos elementos son los siguientes:

“element forceBeamColumn $ eleID $iNode $jNode $transfTag "$IntegrationType, $SecTag, $np"”. Este tipo de elemento requiere que se le asigne una etiqueta ($eleID), de que nodo hasta que nodo se lo define ($iNode $jNode), la transformación geométrica ($transfTag), la cual puede ser lineal, P-delta o Corotacional, acto seguido se deben ingresar los argumentos entre comillas, tales como, tipo de integración ($Integration Type) la cual puede ser Gauss-Lobatto, Gauss-Radau, Legendre o Newton-Cotes, en este documento se usará Gauss-Lobatto. A continuación, una etiqueta que identifique la sección y sus propiedades ($SecTag). Y finalmente el número de puntos de integración ($np). Es necesario respetar el orden para que no exista error.

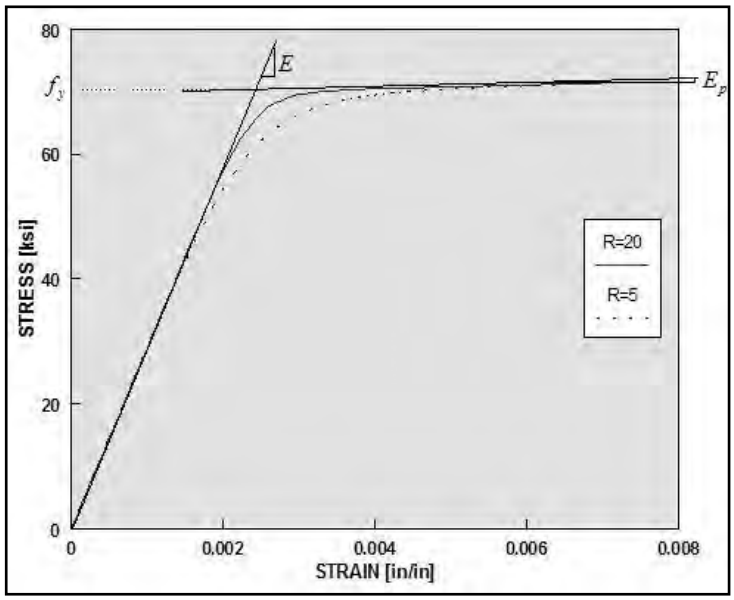
“element forceBeamColumn $tag $ndI $ndJ $transfTag “HingeRadau $secTagI $lpI $secTagJ $lpJ””. Este elemento se lo define para usar la distribución Gauss-Radau modificado, el cual es similar al elemento anterior exceptuando que los argumentos necesarios (sección entre comillas) son, la sección y material asociado a la longitud de rótula plástica inicial ($secTagI), seguido de la longitud de rótula plástica inicial ($lpI), de igual manera es necesario definir el material de la sección y la longitud de rótula plástica ($secTagJ $lpJ). En el medio no será necesario definir ningún parametro, debido a que la sección generalmente es elástica. Este elemento solo se lo usará en un modelo aparte, similar al de “section depth”. Puesto que su definición difiere bastante del modelo convencional.

“element elasticBeamColumn $eleID $iNode $jNode $A $E $I $transfID”. Para los elementos elásticos de igual manera se requiere una etiqueta, nodos que comprenden el elemento, el área de la sección, el módulo elástico, la inercia y la transformación geométrica. “element truss $eleID $iNode $jNode $A $materialID”. De igual manera los elementos tipo truss necesitan un tag, nodos extremos del elemento, área, y una etiqueta en la cual se defina el material. Para estos elementos el material será uno uniaxial, en el cual solo se debe especificar el módulo elástico de la sección (uniaxialMaterial Elastic).

Lo que corresponde a materiales y secciones se debe explicar de forma conjunta, pues las secciones usan los datos de materiales para realizar la división y asignarle el tipo de material. En este modelo se utilizarán varias propuestas de materiales, siendo entre estas las de steel02 (ver Figura 4.5), Hysteretic (ver Figura 4.6) y material en paralelo (ver Figura 4.7) entre Steel02 e hysteretic (ver Figura 4.8). Estos materiales el programa Opensees los tiene definidos, por lo cual solo hay que establecer los datos requeridos. Las divisiones en las secciones se las hará según el proceso de Opensees definido por Lisseth Martinez y Sebastián Pozo en su documento de trabajo de titulación. En el cual se establece dividir las alas en 5 elementos y el alma en 10 elementos (ver Figura 4.9). De igual manera los resortes de los paneles zonales que representaran la flexibilidad del elemento y por ende en la estructura, se los definirá como un material en paralelo de dos steel01 (ver Figura 4.10), cada uno con sus correspondientes datos, para así poder tomar en cuenta el efecto de la placa de refuerzo que el modelo considera. “Este modelo fue propuesto por Krawinkler y publicado en el FEMA 355C” (Martinez Pesantez & Pozo Ocampo, 2018). Es necesario aclarar que el material en paralelo ha sido calibrado solo para columnas, por lo que en los modelos de RBS con fibras, se hará una distinción entre las fibras de las columnas y las fibras de las vigas. Dicha modificación se la realizará dentro del código de Matlab.

Figura 4.5

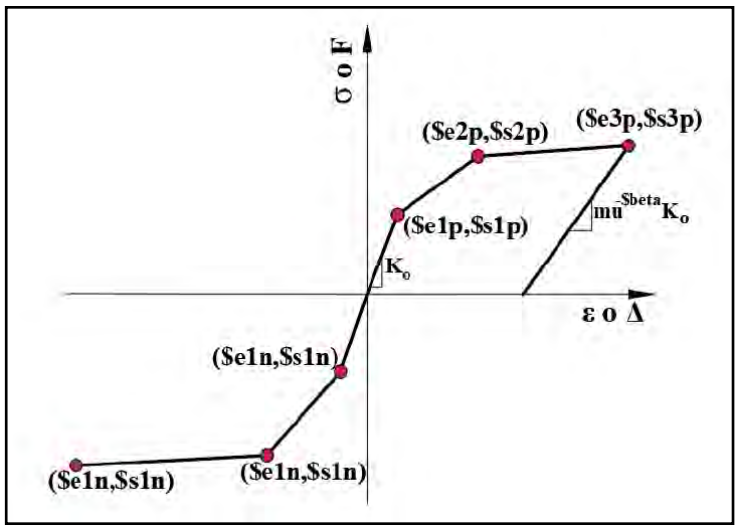
Constitutiva del material steel02



Nota. La fluencia del material se da de manera suavizada. Tomado de (University of Berkeley, 2006)

Figura 4.6

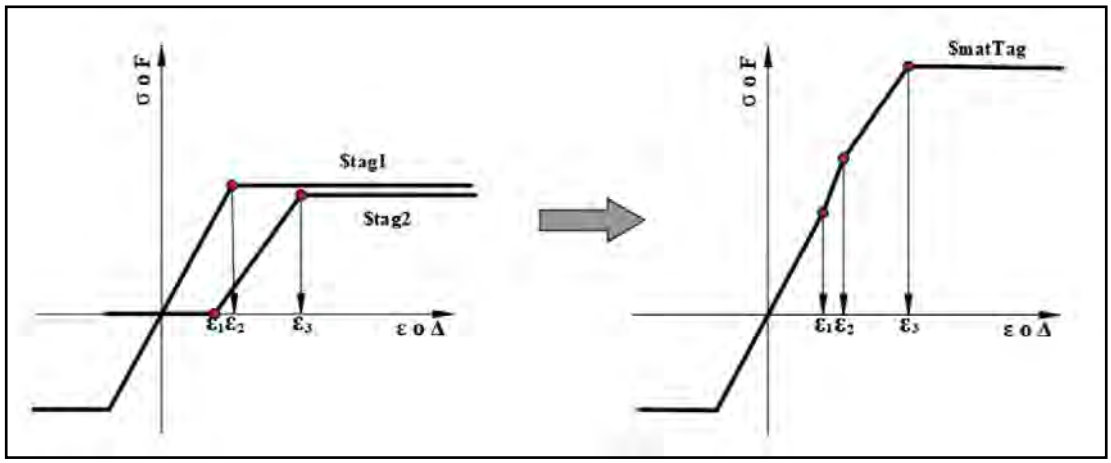
Constitutiva del material Hysteretic



Nota. La curva se construye en base a puntos. Tomado de (University of Berkeley, 2006)

Figura 4.7

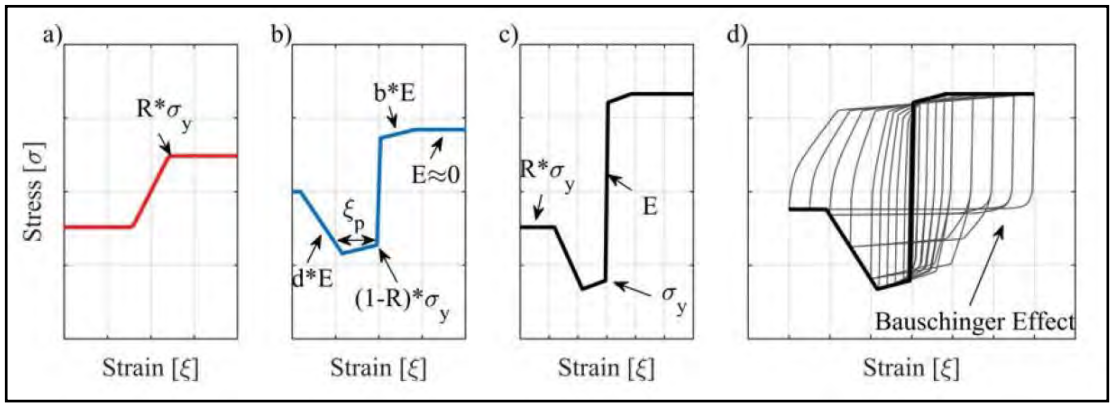
Constitutiva del material en paralelo



Nota. Solo se suman las rigideces y los esfuerzos Tomado de (University of Berkeley, 2006)

Figura 4.8

Constitutiva del material en paralelo steel02 e hysteretic



Nota. a) steel02, b) hysteretic, c) envolvente monotónica, d) envolvente cíclica. Tomado de (Pozo S. , Astudillo, Samaniego, & Flores, 2020)

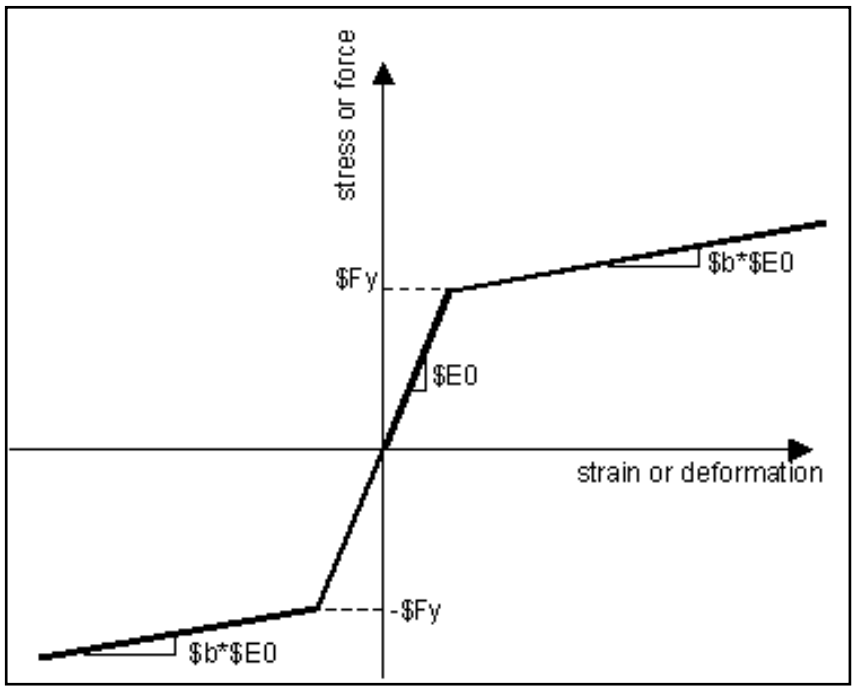
Figura 4.9

Discretización de la sección

Nota. El área equivalente toma en cuenta la curvatura entre el ala y el alma de las secciones W. Tomado de (Martinez Pesantez & Pozo Ocampo, 2018)

Figura 4.10

Constitutiva del material steel01



Nota. El material es bilineal y solo toma en cuenta el endurecimiento por deformación (b). Tomado de (University of Berkeley, 2006)

Los resortes en este modelo solo se presentarán en los paneles zonales y en las bases, utilizando en las bases materiales uniaxiales con módulos muy altos para tener en cuenta el empotramiento que causa la cimentación. El Opensees reconoce estos resortes como elementos "zerolength" y su ingreso se realiza de la siguiente manera:

"element zeroLength Tag node i node j -mat matTag -dir #". Se requiere definir la etiqueta del elemento, los nodos que lo comprenden, la etiqueta del material, y en que dirección está actuando este material. Por consiguiente, es necesario conocer los grados de libertad con que se está trabajando en la estructura. En este caso puesto que el modelo es en el plano cada nodo presentaría 3 grados de libertad (desplazamiento en X, desplazamiento en "Y" y rotación alrededor del eje Z). Sin embargo, el programa mantiene la misma numeración como si fuera tridimensional. (1 para desplazamiento en X, 2 para desplazamiento en Y, 3 para desplazamiento en Z, 4 para rotación alrededor del eje X, 5 para rotación alrededor del eje "Y" y 6 para rotación alrededor del eje Z).

Los constraints son las condiciones que establecerán una disminución de los grados de libertad mediante el comando "equal dof", en el cual solo se necesita definir el tag de los nodos y los grados de libertad que se quieran igualar. Esto se lo usará para darle continuidad a las columnas con splices y a los paneles zonales (prácticamente todos los nodos que tienen exactamente las mismas coordenadas). También se utilizará este comando para asignar los diafragmas de piso, igualando los grados de libertad de desplazamiento en sentido X de los nodos centrales derechos de los paneles zonales (xy005), adjuntando a estos los nodos de la "leaning column".

Las masas sísmicas de la estructura se las colocarán en los nodos centrales derechos de los paneles zonales (xy005), y sus valores son los correspondientes al peso de cada piso. Se

eligen estos nodos ya que son los principales medios por los que se distribuyen los esfuerzos a las columnas.

En este punto se tendría definido el modelo de “fiber hinge”. Como se puede intuir, para establecer este modelo se necesita ingresar de manera repetitiva una gran cantidad de datos, por lo que es necesario el uso del Matlab para generar estos archivos. Esto también ayuda con el hecho de que se hace más fácil corregir cualquier error, ya que solo se modifica el código y se lo vuelve hacer correr, modificándose así de manera automática los valores solicitados.

Para una revisión detallada del código y los archivos generados, se recomienda revisar los Anexos.

4.2. Modelos de Análisis

A continuación, se procede a generar un modelo de plasticidad distribuida, para comparar su comportamiento con el propuesto anteriormente. En este modelo solo basta con cambiar los elementos elásticos (en columnas) por elementos tipo fibra controladas por fuerza (element forceBeamColumn). Este tipo de ejemplar es parecido al propuesto por Pozo, Astudillo, Samaniego y Flores en su documento “Objective Phenomenological Constitutive Law for Collapse Analyses in 1 Distributed Plasticity Steel Frame Models”. Sin embargo tiene una variación en la definición de la longitud de rótula plástica, ya que se conserva el L/12 y no se utiliza la altura de la sección (como indica el documento).

También se realizará un modelo en el cual la zona de RBS se la definirá como plasticidad concentrada, mientras en las columnas se mantendrá las fiber hinges en la longitud de L/12. Estos resortes se los modelará siguiendo los parametros de los estudios propuestos por Ibarra, Medina y Krawinkler (mod IMK), ya que son capaces de capturar el deterioro debido al pandeo en las alas o patines de la sección ya sea por pandeo local o pandeo lateral torsional (LTB). Por último se inspeccionará el comportamiento el modelo con lóngitud de rótula plástica

igual a la altura de la sección de cada columna utilizando distribuciones Gauss-Lobatto y Gauss-Radau modicido. (Pozo S. , Astudillo, Samaniego, & Flores, 2021 en revisión)

4.3. Ejecución de Modelos

Se harán tres tipos de ejecuciones, gravitacional para conocer los periodos, las fuerzas y las reacciones en los elementos. Análisis Pushover para obtener la capacidad de las diferentes estructuras. Y dinámico para observar el comportamiento de los diagramas momento-curvatura. Esta parte se la realiza codificando directamente en un archivo TCL, dado que no es iterativa no hay necesidad ni ventaja de programar en Matlab.

Para realizar el análisis gravitacional se debe introducir el modelo haciendo “source” a los archivos TCL, luego se deben asignar las cargas gravitacionales de los frame y de la leaning column. Finalmente es necesario definir los parámetros de la corrida del análisis. Debido a que se usará el algoritmo de Newton para resolver el sistema de múltiples grados de libertad, es necesario definir una tolerancia, la cual se usará $1 * 10^{-6}$. Para los constraints se usará “plain” ya que se está trabajando en 2D. El Opensees da la opción de minimizar los grados de libertad usando “numberer RCM”. El proceso matemático será resuelto por medio de “System BandGeneral”. Sin embargo, no siempre se puede usar este método. Todo depende de la complejidad de la estructura, por lo que es necesario estar atento a los mensajes de error que involucren a las matrices de masa y rigidez. Mediante “test NormDispIncr” se chequea si se logró cumplir con la tolerancia la final de la interación. Usando “NstepGravity” y “DGravity” se define la cantidad de pasos necesaria para que la carga se aplique completamente (se está aplicando la carga de manera gradual), y el incremento de la carga, respectivamente. Por medio de “integrator LoadControl” se establece siguiente paso en el tiempo. Al definir “analysis Static” se indica al programa que las cargas de demanda son estáticas y no transientes. Finalmente se ingresa “analyze” para ejecutar el análisis. Con el comando “loadConst -time

0.0” se le indica al Opensees que la carga gravitacional la mantenga, pero que regrese el tiempo a 0 (lo cual deja el análisis listo para el pushover). Para finalizar con el análisis gravitacional es necesario conocer los periodos de la estructura. Por lo tanto, se realiza el proceso de determinar los valores propios del sistema de ecuaciones, estos valores propios el programa los calcula mediante el comando “genBandArpack”. Sin embargo, al igual que “System BandGeneral” depende del sistema de ecuaciones, por lo que no siempre funciona correctamente y se lo debe cambiar. Una vez realizado este proceso se obtienen las frecuencias de vibración elevadas al cuadrado, por lo cual es necesario realizar la correspondiente operación matemática y transformación para así obtener los periodos de la estructura.

Las otras alternativas para la resolución de los valores propios son “symmBandLapack” y “fullGenLapack”. De igual manera para resolver el sistema de ecuaciones se tiene “BandSPD”, “ProfileSPD”, “SuperLU”, “UmfPack”, “FullGeneral”, “SparseSYM” y “Cusp”. Debido a la programación de estos procesos suele haber errores en las resoluciones, por lo que se debe probar con las diferentes alternativas. Para este proyecto se determinó que las indicadas anteriormente fueron satisfactorias.

El análisis pushover se lo realizará a continuación del análisis gravitacional, por lo tanto se puede copiar el código anterior, pegarlo en un nuevo archivo y codificar a partir de ese. Primero es necesario definir los datos que se requieran grabar para posteriormente analizarlos, para esto se utilizará el comando “recorder”. Mediante este comando se pueden obtener datos ya sean de nodos o elementos. En este proyecto se capturará los datos necesarios para analizar las capacidades de las distintas estructuras (fuerzas en las columnas y desplazamientos en el nodo de techo). A partir de este punto es necesario definir las cargas laterales, las cuales se las calculará en base al ASCE 7-10. Acto seguido se necesita introducir los parámetros de análisis del pushover con respecto al desplazamiento. Estos parámetros son los del nodo de control y el

grado de libertad asociado, el desplazamiento máximo y la razón de incremento. Esto mediante los parámetros “IDctrlNode”, “IDctrlDOF”, “Dmax”, “Dincr”, respectivamente. Luego es necesario definir los parámetros de análisis. Dichos parámetros son iguales a los de la parte gravitacional (“constraints Plain”, “numberer Plain”, “system UmfPack”, tolerancia), solo hay que agregar el máximo número de interaciones (“maxNumIter”) y el método para chequeo de convergencia, el cual se usará “TestType EnergyIncr” y “NormUnbalance”. Finalmente hay que definir el algoritmo a usar, el cual será el algoritmo de Newton (“algorithm Newton”). Para que el programa ejecute el algoritmo de newton, es necesario especificar el control de desplazamientos mediante “integrator DisplacementControl”, y que el análisis sea con cargas de demanda estáticas (“analysis Static”). El número de pasos “Nsteps” será la razón entre el desplazamiento máximo y el incremento. Debido a que las convergencias en los modelos matemáticos son la parte más complicada, usará otro algoritmo adicional, el cual deliberadamente hará más pequeño el incremento y utilizará otros algoritmos de convergencia. Si al final del uso de estos no existe convergencia, se indicará presentar un mensaje “PushOver failed at step”. Este archivo se lo nombrará “PO_convergence_loop.tcl”.

Para el análisis dinámico es necesario tener definido previamente un registro o varios registros dependiendo del análisis. Para este proyecto solo se seleccionó un registro, el de chichi Taiwan, debido a que su duración (52.79 segundos) ayuda a obtener una mayor cantidad de información sobre el comportamiento de las fibras, y computacionalmente se lo considera óptimo. Otro punto para destacar es que el registro estese filtrado, es decir que la señal no presente ninguna perturbación ambiental. El registro debe encontrarse en formato “TXT” y solo debe constar con los valores de aceleración en función de g. Para que el Opensees lo cargue, se debe crear un “TXT” con los siguientes datos: Tag del registro, Nombre del registro (Nombre del archivo “TXT” de las aceleraciones), número de puntos, intervalo de tiempo y factor de

escala. Al igual que en el pushover es necesario realizar previamente un análisis gravitacional, por lo que se puede de igual manera copiar el análisis gravitacional (realizado anteriormente) y pegarlo en nuevo archivo “TCL”. En el análisis dinámico se deberá definir el amortiguamiento de la estructura mediante el método de Rayleigh (que utiliza el primer y tercer periodo de vibración) en base a un porcentaje de amortiguamiento crítico de 2.5%. Los resultados se grabarán mediante el comando “recorder” y serán de los elementos tipo fibra, sus momentos y rotaciones. Aparte también se grabarán los resultados de aceleraciones y derivas de piso. En los parámetros de resolución del método numérico se usará de igual manera el algoritmo de Newton. Sin embargo, se usará un análisis transiente y no estático debido a que se introducirá un movimiento en la base. Como solver se ejecutará el método de Newmark (“integrator Newmark”) y al igual que en el análisis pushover se deberá proporcionar procesos de convergencia en un archivo aparte, para este ejemplar se lo denominará “GM_convergence_loop.tcl”.

En este punto el análisis se lo realizó en un edificio de 8 pisos. Sin embargo, como se utiliza el Matlab para generar el modelo, solo es necesario cambiar número de pisos para generar los otros modelos concernientes a la investigación. La codificación de las ejecuciones de los modelos no varía, excepto por el tag de los elementos o nodos en los recorders que se desee obtener los resultados. De igual manera es necesario ejecutar los pushover en los edificios hasta que tengan una pérdida de resistencia considerable, para que se produzca el colapso. También es necesario generar un código en Matlab para postproceso de los resultados, debido a que los resultados provistos en el Opensees son solamente numéricos. Mediante esta Herramienta en conjunto con el Excel, se generarán los gráficos necesarios para observar el comportamiento de las estructuras propuestas.

CAPITULO 5.- ANÁLISIS DE RESULTADOS

5.1. Análisis Estático No lineal

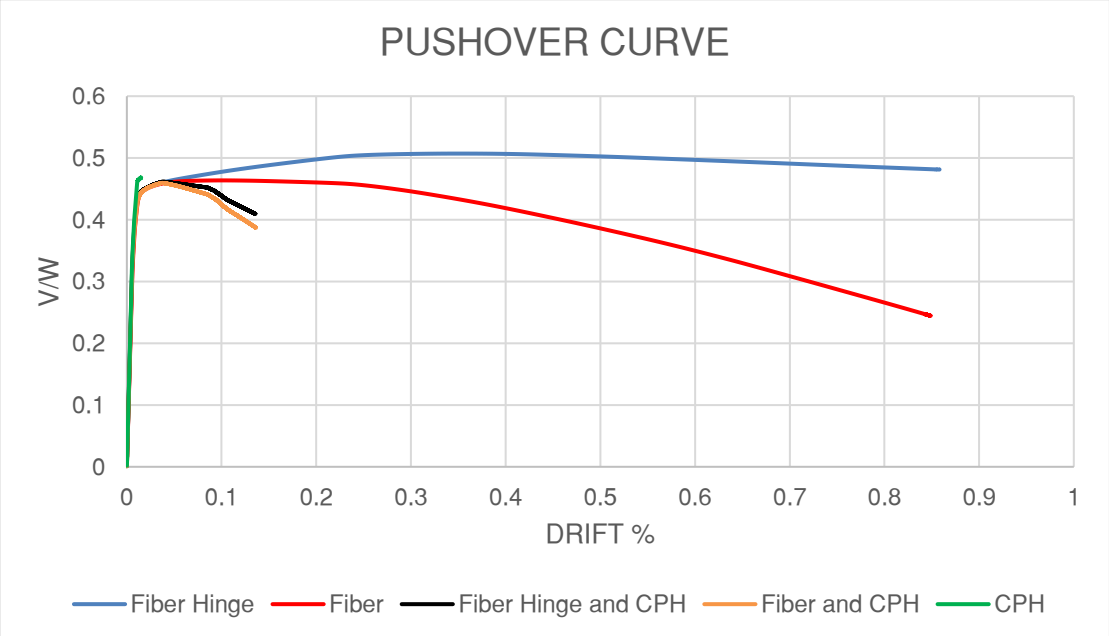
Los datos de las curvas CPH (resortes en vigas y columnas) fueron tomados de Torres-Rodas et al., 2020, por motivos de comparación y análisis.

5.1.1. Resultados con Material Steel02

Los gráficos a continuación muestran el comportamiento de las estructuras de diferentes pisos, con una longitud de rótula plástica de L/12 seccionada en tres elementos (donde L es la altura de piso), para los diferentes tipos de modelos.

Figura 5.1

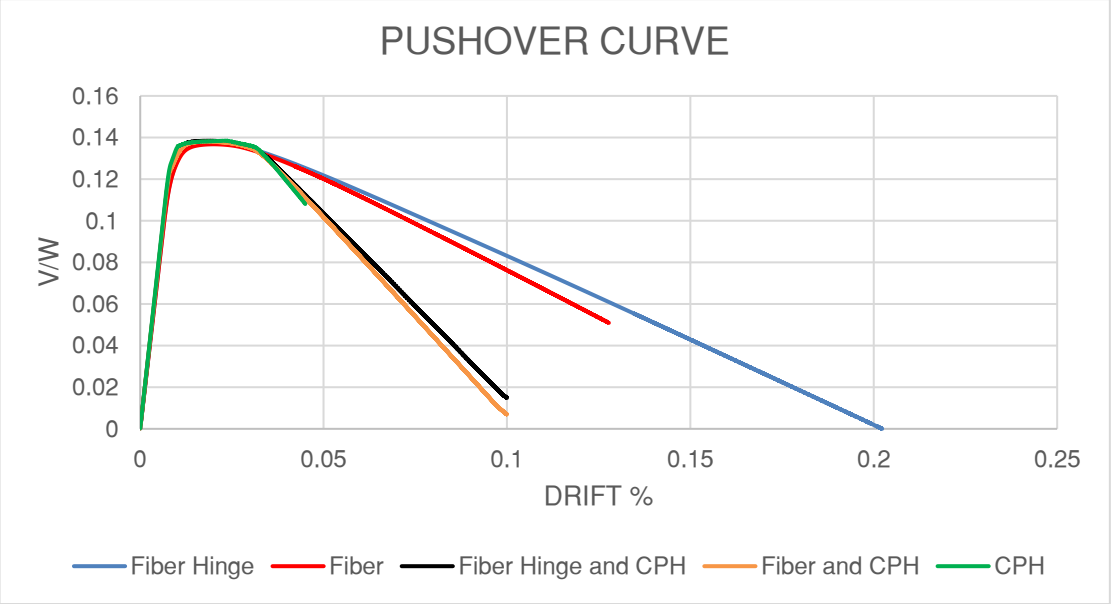
Curvas Pushover del edificio de 2 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.2

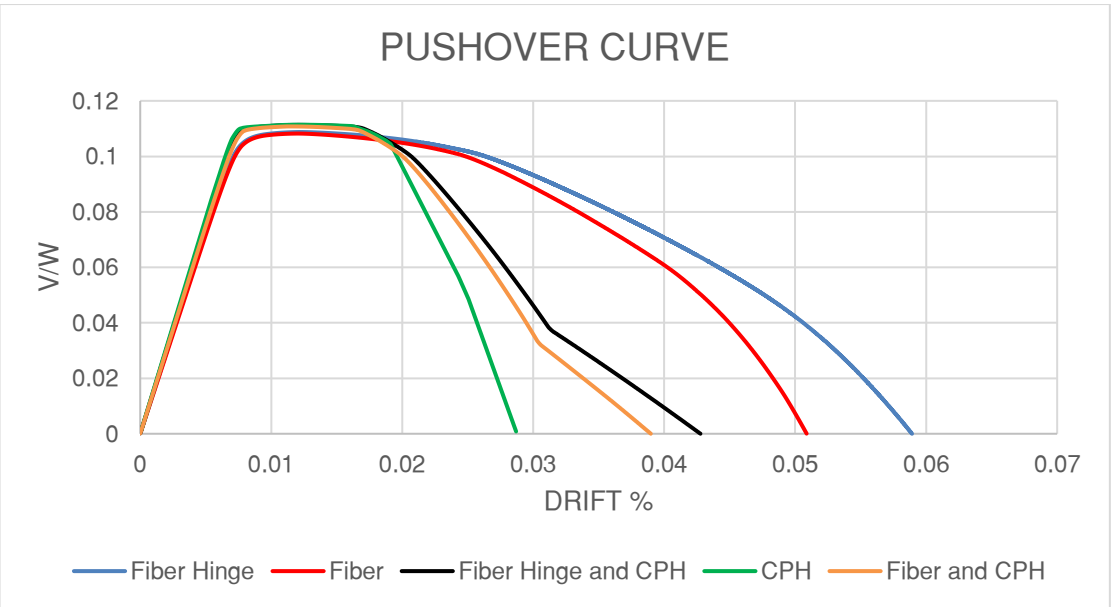
Curvas Pushover del edificio de 4 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.3

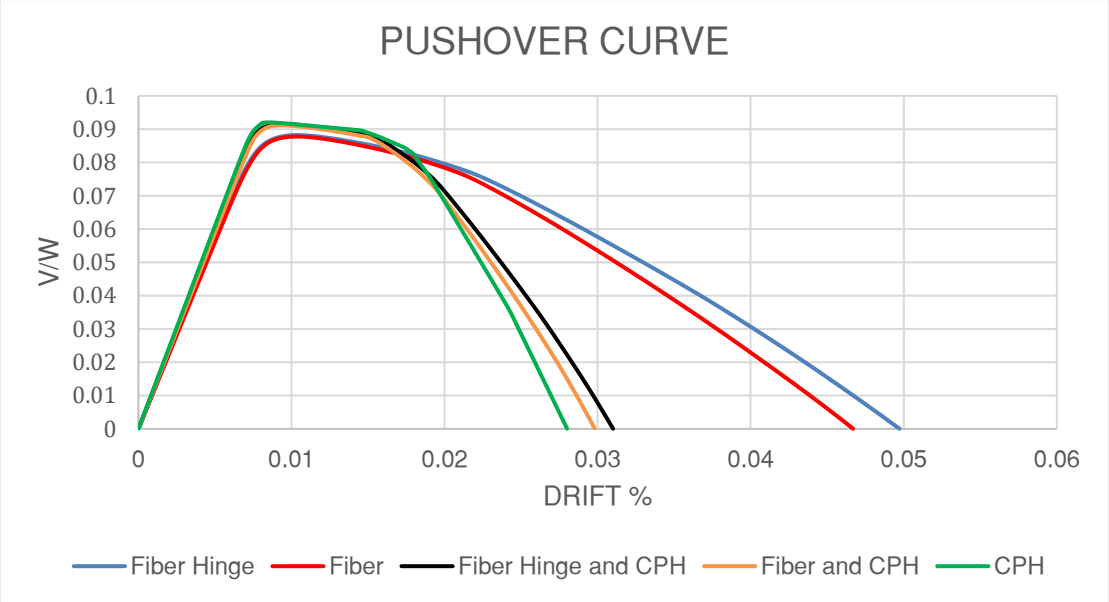
Curvas Pushover del edificio de 8 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.4

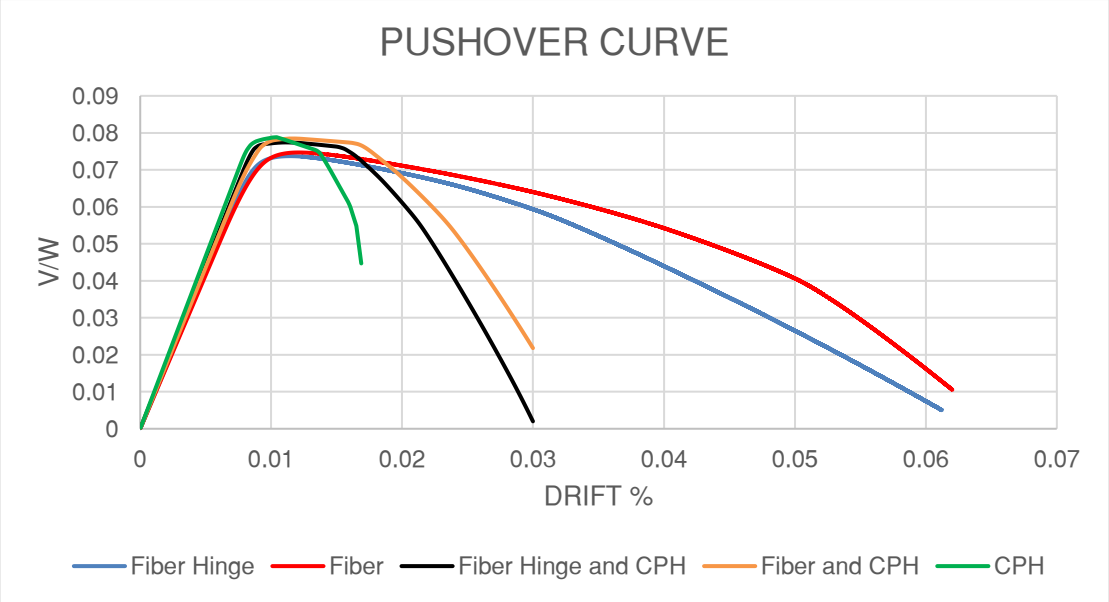
Curvas Pushover del edificio de 12 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.5

Curvas Pushover del edificio de 20 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Se observa que los modelos que poseen una mayor efectividad en la captura del deterioro de la capacidad de la estructura son los “Fiber Hinge and CPH” y “Fiber and CPH” (Debido a que su curva se acerca mucho a la de CPH). El modelo “Fiber and CPH” corresponde al de plasticidad distribuida en toda la columna y resortes en las vigas, mientras que el “Fiber Hinge and CPH” posee plasticidad distribuida solo en la zona de rótula plástica.

Se logra apreciar que el edificio de 2 pisos muestra un comportamiento prácticamente elástico, debido a que su deterioro se produce a un nivel muy alto de deriva (cerca del 10%). En cambio, en el edificio de 20 pisos se evidencia una mediana diferencia entre los modelos “CPH” y fibras con rótulas.

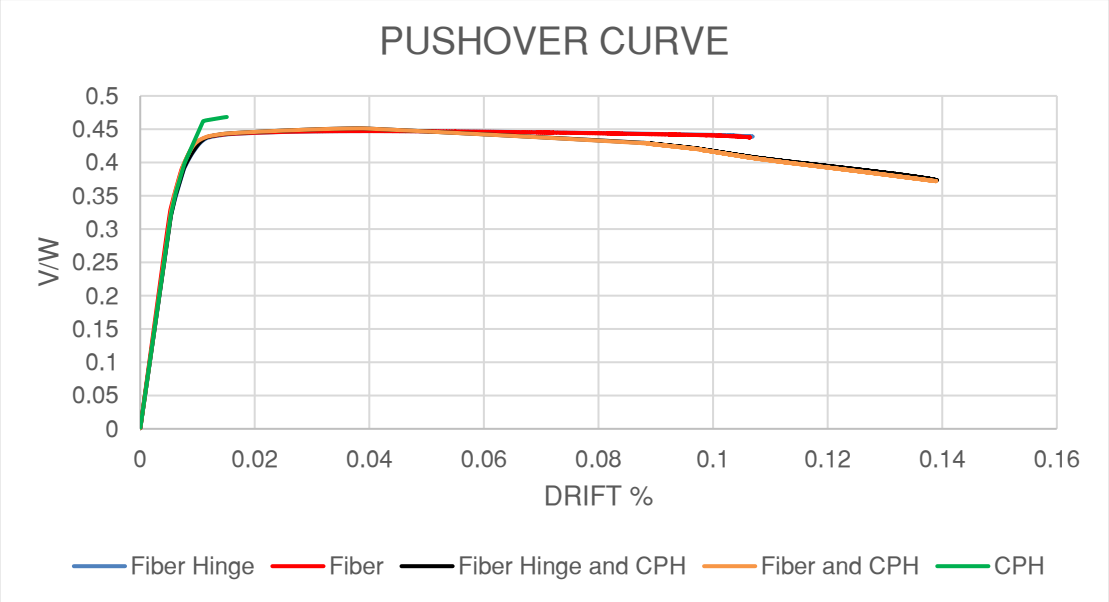
También llama la atención el hecho que los modelos que poseen fibras tanto en vigas como columnas (Fiber Hinge y Fiber), no son capaces de capturar el deterioro adecuadamente. En gran parte esto se produce porque las fibras en las rótulas en las vigas (zonas RBS) no alcanzan a deteriorarse.

5.1.2. Resultados con Material Hysteretic

En los siguientes gráficos se mantuvo la longitud de rótula plástica, sin embargo se cambió el material Steel 02 por Hysteretic. De igual manera se incluyó la curva CPH para comparar su comportamiento.

Figura 5.6

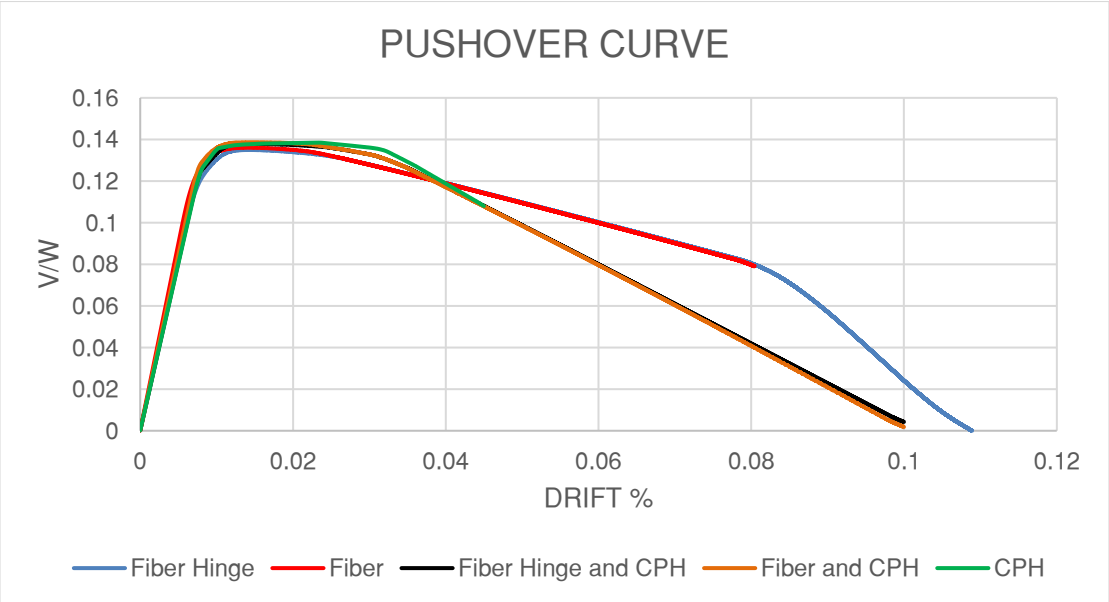
Curvas Pushover del edificio de 2 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.7

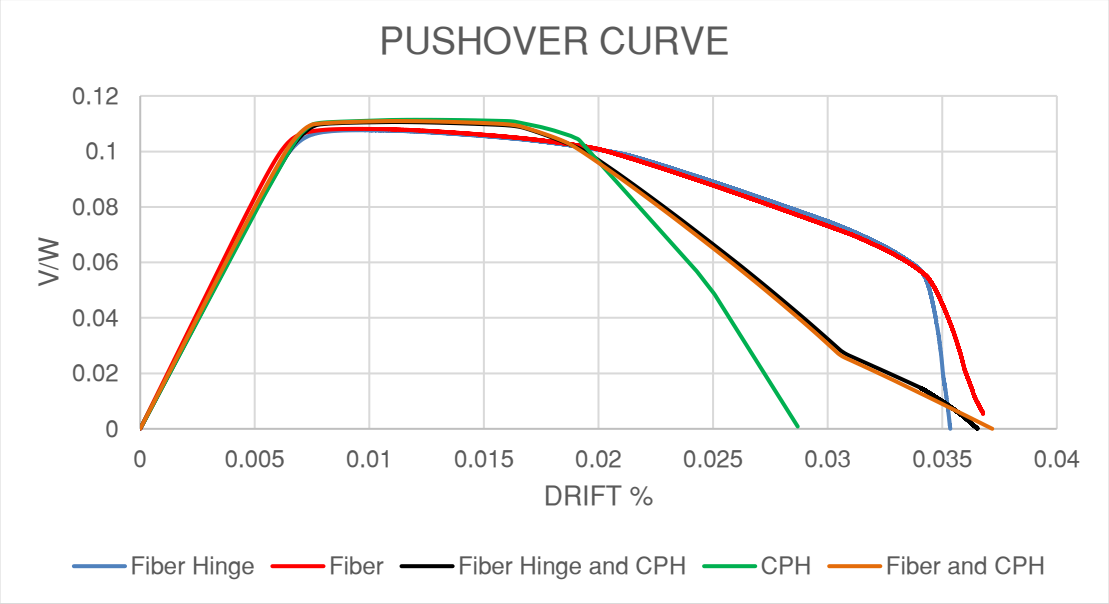
Curvas Pushover del edificio de 4 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.8

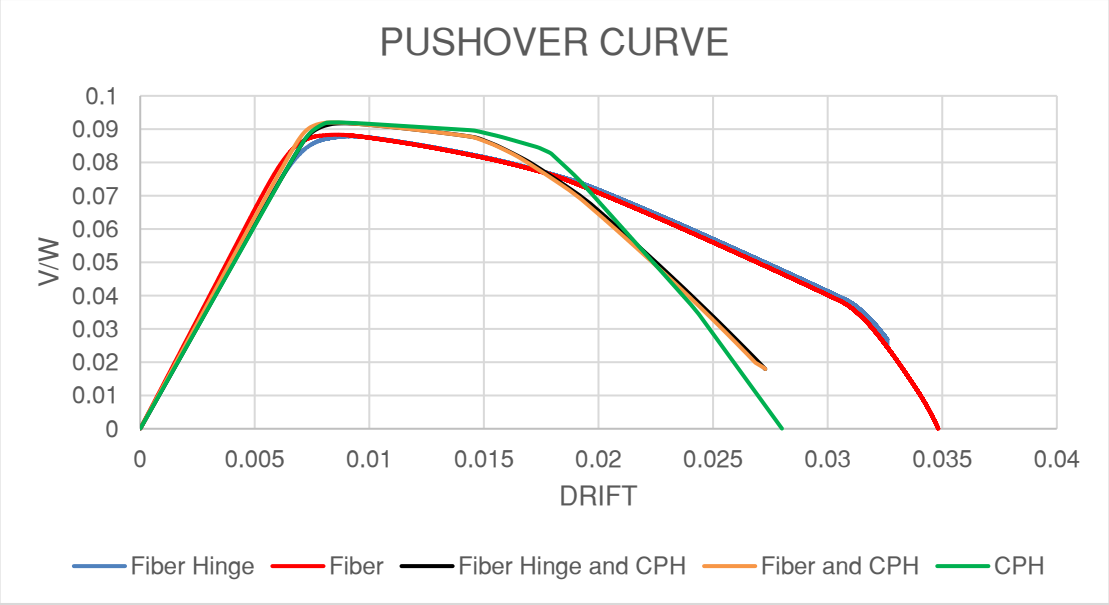
Curvas Pushover del edificio de 8 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.9

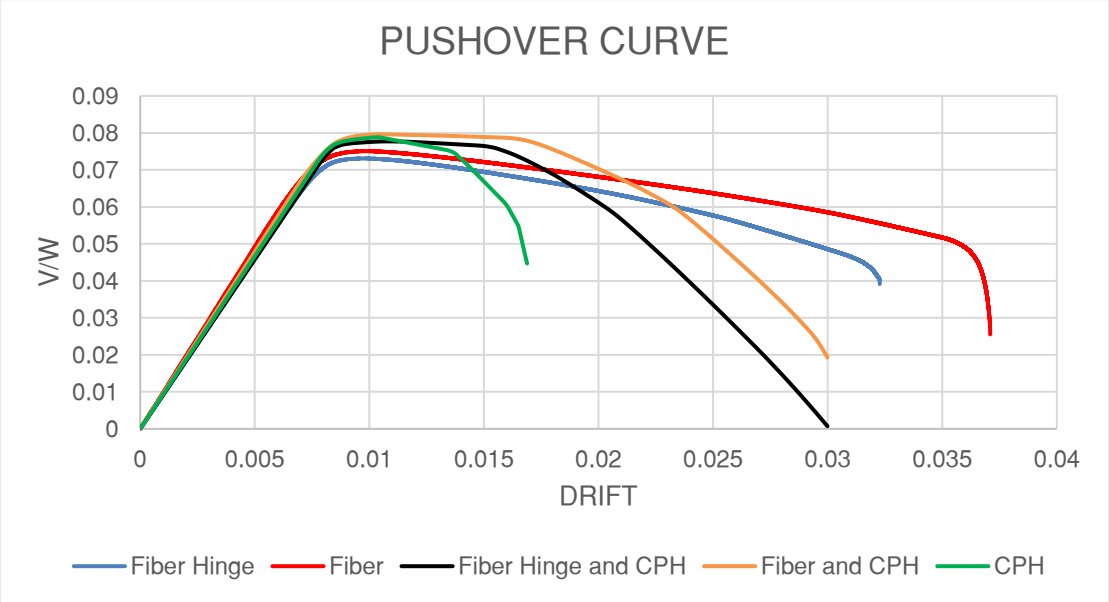
Curvas Pushover del edificio de 12 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.10

Curvas Pushover del edificio de 20 pisos



Nota. La curva CPH corresponde a "Concentrated Plastic Hinge".

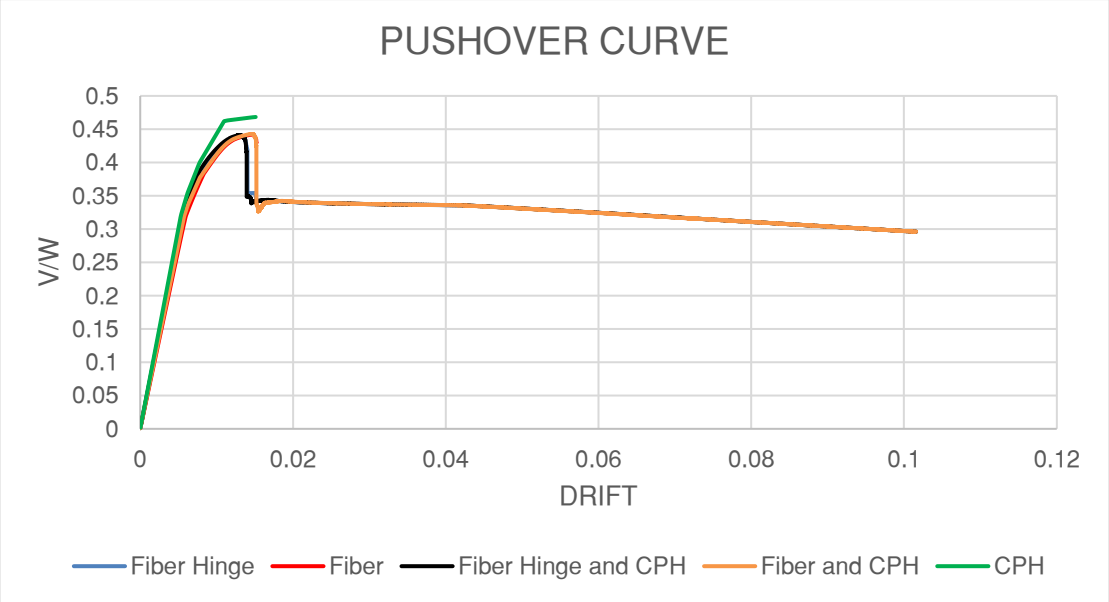
Con el material Hysteretic se puede observar problemas en la convergencia, enfocado especialmente en los modelos "Fiber Hinge" y "Fiber". Sin embargo, nuevamente se logra apreciar una afinidad (hacia el modelo CPH) entre los modelos con resortes en vigas.

5.1.3. Resultados con Material Parallel (Hysteretic & Steel02)

Los datos del material en paralelo entre Hysteretic y Steel02 fueron tomados de Pozo et al.,2021 en revisión. También hay que tener en cuenta que este material solo fue propuesto para columnas, por lo que el material para las vigas en los modelos Fiber Hinge y Fiber se utilizara Steel02.

Figura 5.11

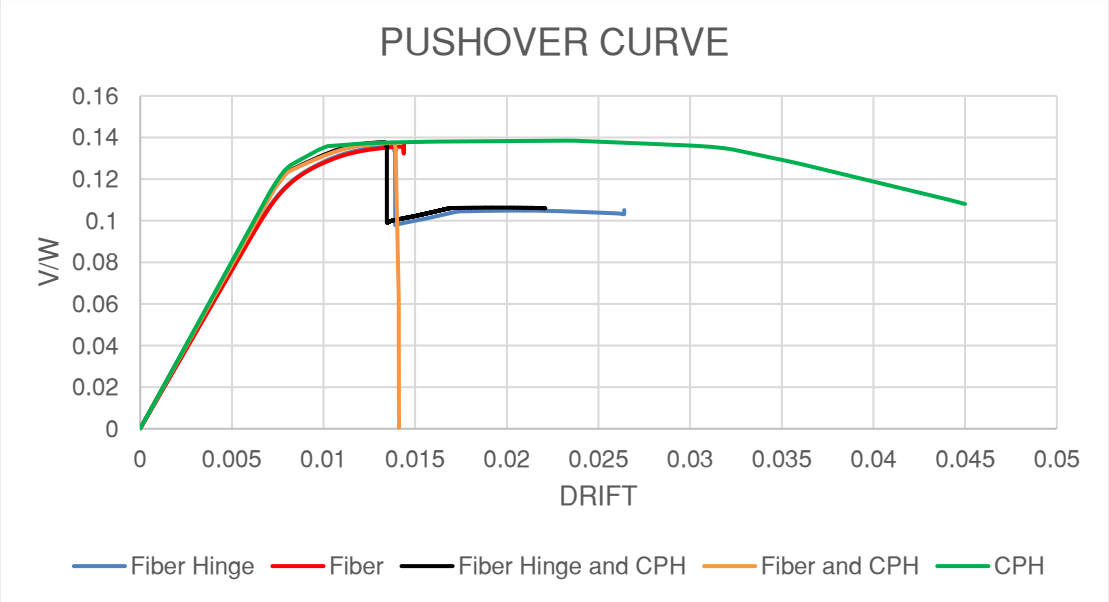
Curvas Pushover del edificio de 2 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.12

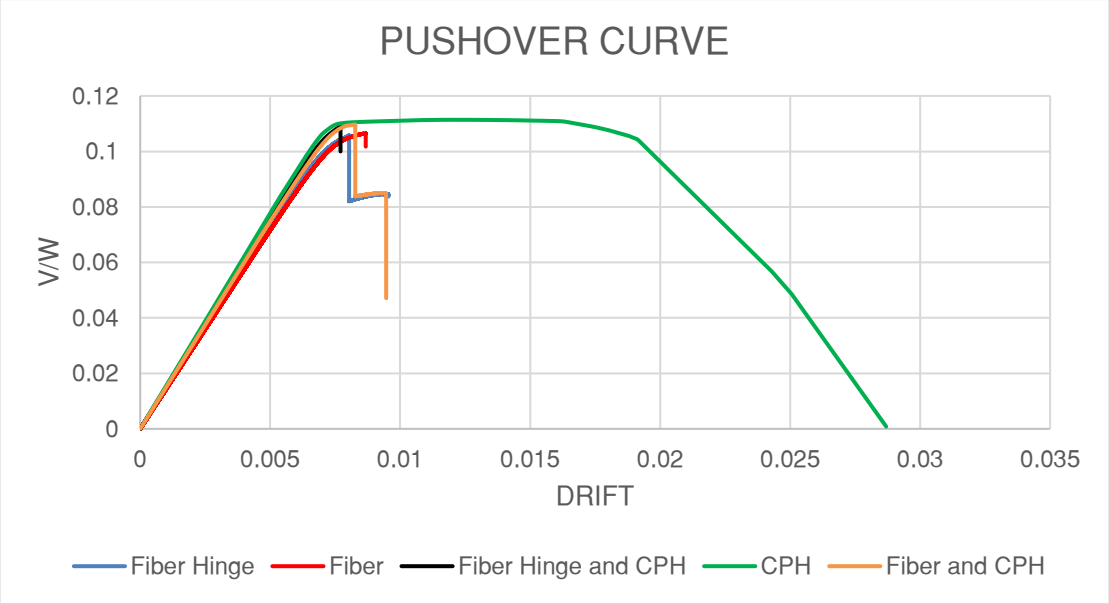
Curvas Pushover del edificio de 4 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.13

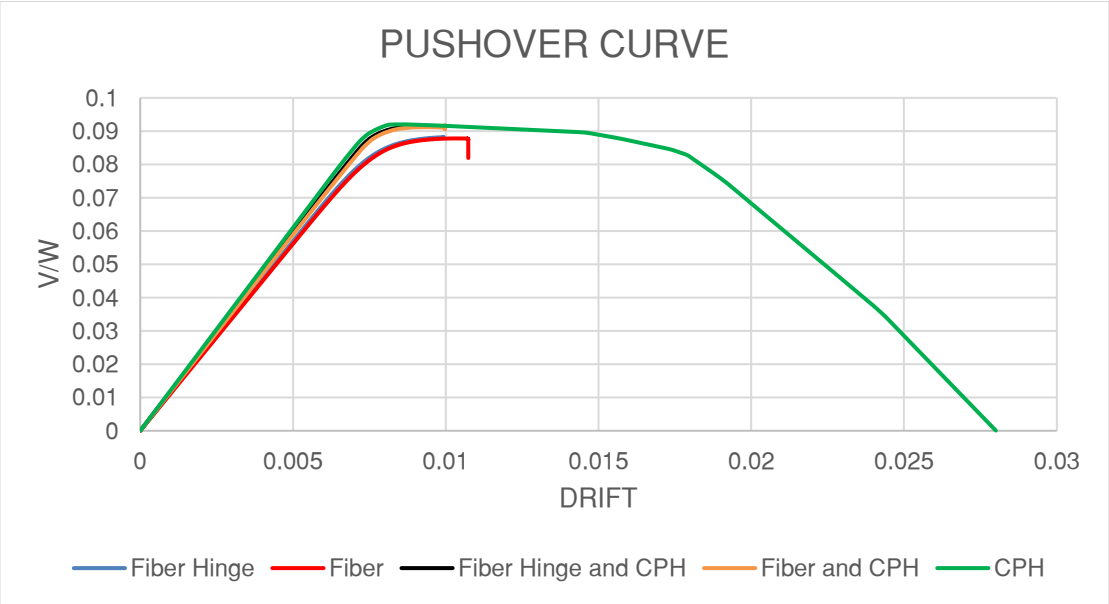
Curvas Pushover del edificio de 8 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.14

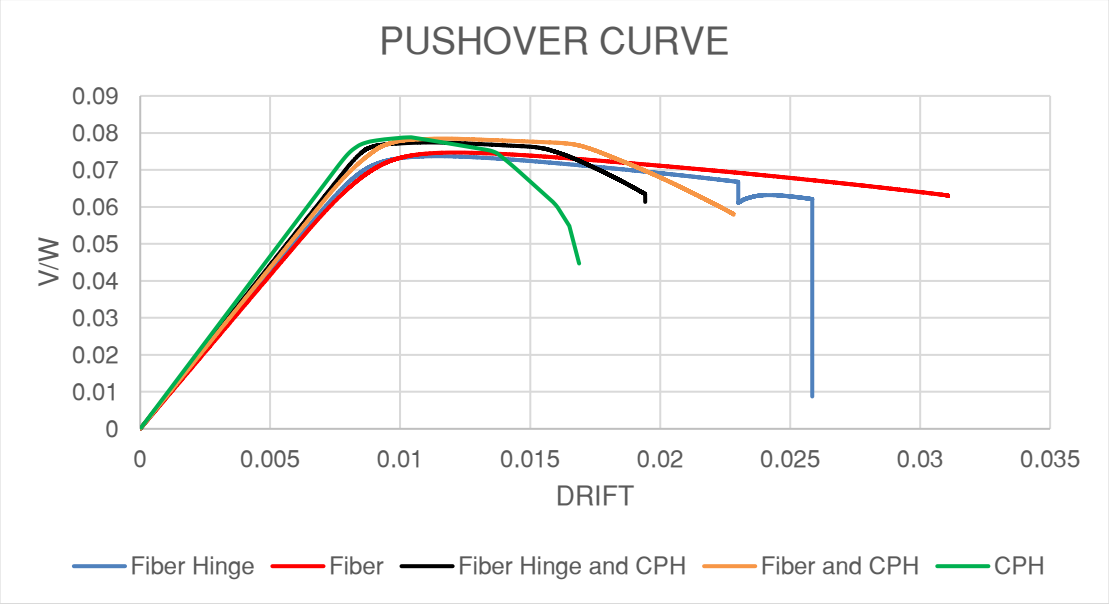
Curvas Pushover del edificio de 12 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.15

Curvas Pushover del edificio de 20 pisos



Nota. La curva CPH corresponde a "Concentrated Plastic Hinge".

Es necesario tener en cuenta que el material en paralelo está orientado solo para los elementos tipo columna. Sin embargo, en los modelos que utilizan fibras en las vigas, se usó material Steel02.

Con el material en paralelo se puede apreciar incluso más que con el material Hysteretic, la dificultad para converger de los modelos. Dado que ningún modelo posee una curva completa.

También se analizará el comportamiento de los modelos en bases de eficiencia computacional, y variación de los periodos en las tablas siguientes.

Tabla 7

Propiedades de los modelos con material Steel02

# Pisos	Modelo	T (s)	ΔX (in)	X (% de H)
2	Fiber Hinge	0.649	0.1	90
	Fiber	0.663	0.1	90
	Fiber Hinge and CPH	0.640	0.005	60
	Fiber and CPH	0.653	0.005	60
4	Fiber Hinge	1.653	0.1	90
	Fiber	1.649	0.1	90
	Fiber Hinge and CPH	1.627	0.005	10
	Fiber and CPH	1.629	0.005	10
8	Fiber Hinge	2.287	0.05	50
	Fiber	2.312	0.1	30
	Fiber Hinge and CPH	2.238	0.1	10
	Fiber and CPH	2.261	0.1	10
12	Fiber Hinge	3.088	0.1	40
	Fiber	3.101	0.1	40
	Fiber Hinge and CPH	3.014	0.1	10
	Fiber and CPH	3.033	0.1	10
20	Fiber Hinge	4.399	0.05	50
	Fiber	4.396	0.1	50
	Fiber Hinge and CPH	4.286	0.1	3
	Fiber and CPH	4.296	0.1	3

Nota. "H" es la altura total del edificio correspondiente

Tabla 8

Propiedades de los modelos con material Hysteretic

# Pisos	Modelo	T (s)	ΔX (in)	X (% de H)
2	Fiber Hinge	0.627	0.005	90
	Fiber	0.612	0.005	90
	Fiber Hinge and CPH	0.629	0.005	60
	Fiber and CPH	0.623	0.005	60
4	Fiber Hinge	1.595	0.005	90
	Fiber	1.523	0.005	90
	Fiber Hinge and CPH	1.605	0.005	10
	Fiber and CPH	1.568	0.005	10
8	Fiber Hinge	2.199	0.005	50
	Fiber	2.137	0.005	50
	Fiber Hinge and CPH	2.206	0.01	10
	Fiber and CPH	2.181	0.01	10
12	Fiber Hinge	2.967	0.005	50
	Fiber	2.868	0.005	50
	Fiber Hinge and CPH	2.974	0.1	10
	Fiber and CPH	2.922	0.1	10
20	Fiber Hinge	4.223	0.005	50
	Fiber	4.062	0.005	50
	Fiber Hinge and CPH	4.222	0.1	3
	Fiber and CPH	4.115	0.1	3

Nota. "H" es la altura total del edificio correspondiente

Tabla 9

Propiedades de los modelos con material Parallel (Hysteretic y Steel02)

# Pisos	Modelo	T (s)	ΔX (in)	X (% de H)
2	Fiber Hinge	0.649	0.05	30
	Fiber	0.663	0.05	30
	Fiber Hinge and CPH	0.640	0.005	90
	Fiber and CPH	0.653	0.005	60
4	Fiber Hinge	1.653	0.05	30
	Fiber	1.657	0.05	30
	Fiber Hinge and CPH	1.627	0.005	60
	Fiber and CPH	1.629	0.005	60
8	Fiber Hinge	2.286	0.0005	30
	Fiber	2.312	0.0005	30
	Fiber Hinge and CPH	2.237	0.005	50
	Fiber and CPH	2.261	0.005	50
12	Fiber Hinge	3.088	0.005	50
	Fiber	3.110	0.005	50
	Fiber Hinge and CPH	3.014	0.005	50
	Fiber and CPH	3.033	0.005	50
20	Fiber Hinge	4.399	0.005	50
	Fiber	4.413	0.005	50
	Fiber Hinge and CPH	4.286	0.005	50
	Fiber and CPH	4.296	0.005	50

Nota. "H" es la altura total del edificio correspondiente

Se evidencia que los periodos de los elementos son casi similares, tomando en cuenta la variación de material y modelo. Sin embargo, en el material Hysteretic y Parallel se observa un costo computacional importante, debido a que el intervalo de desplazamiento es de 0.005 o incluso 0.0005 en algunos casos, lo que se traduce en una mayor cantidad de pasos en el análisis.

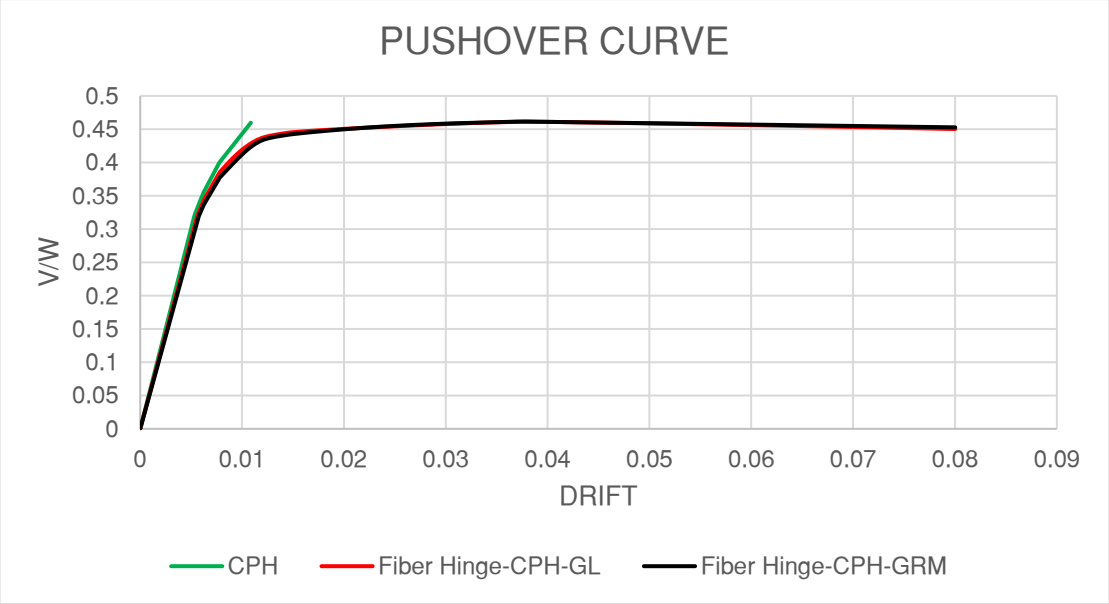
Esta disminución en el intervalo se la realiza para tratar de converger en el modelo. Por lo tanto, se puede comprender que las estructuras con una convergencia complicada usando un material Steel02, son las de 2 y 4 pisos. En los modelos que poseen rótulas en las vigas concretamente.

5.1.4. Resultados Pushover entre modelos GL y GRM

Por consiguiente, se observa que los modelos con mayor estabilidad son los "Fiber Hinge and CPH" y "Fiber and CPH", usando un material Steel02. Entonces se decide utilizar el "Fiber Hinge and CPH" y el material de Steel02 para los análisis restantes. En los modelos siguientes prácticamente se hará la comparativa entre usar puntos de integración tipo Gauss-Lobatto (GL) y Gauss Radau Modificado (GRM). En ambos modelos también se procederá a aumentar el fibrado de la sección (20 fibras en las alas y en el alma), como sugerencia de los autores antes citados. Sin embargo, una diferencia importante es que para los modelos GL se usará una longitud de rótula plástica igual a la altura de la sección de la columna, mientras que para los modelos GRM se usará 3/5 de la altura de la sección.

Figura 5.16

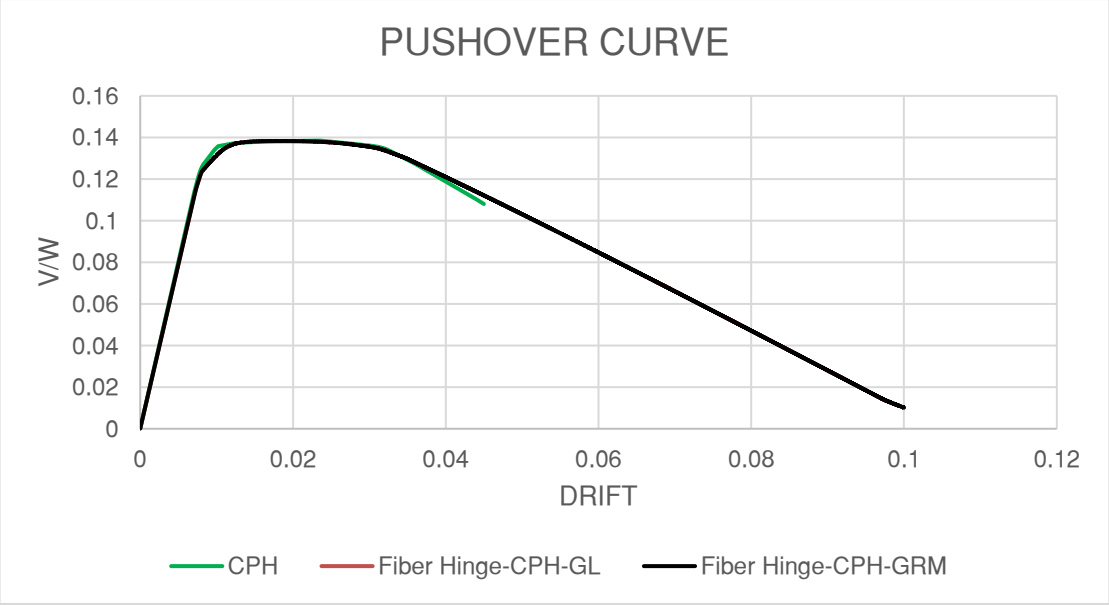
Curvas Pushover del edificio de 2 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.17

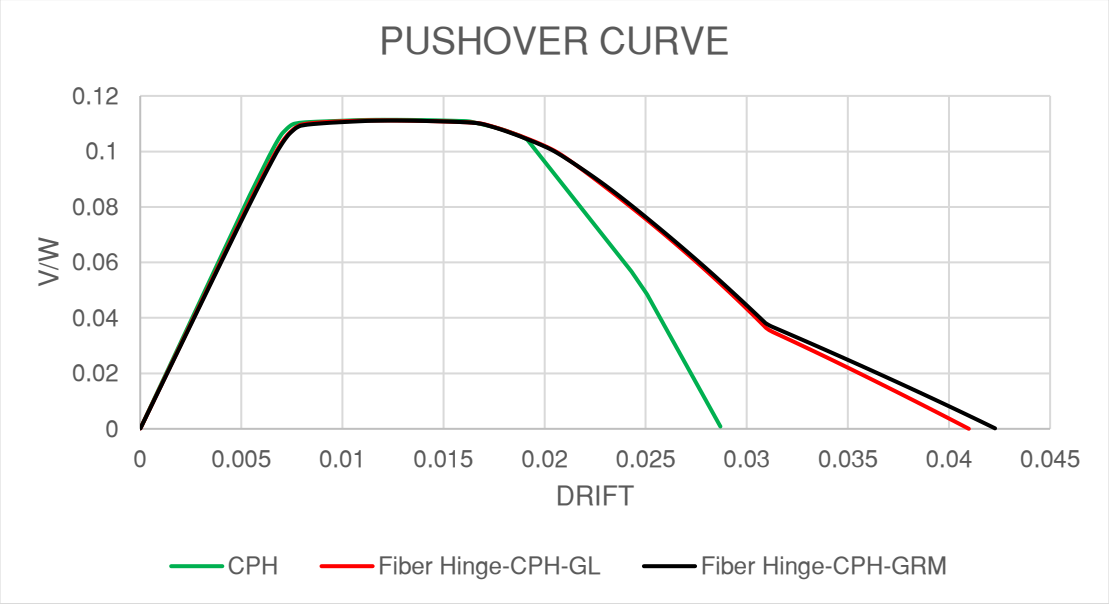
Curvas Pushover del edificio de 4 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.18

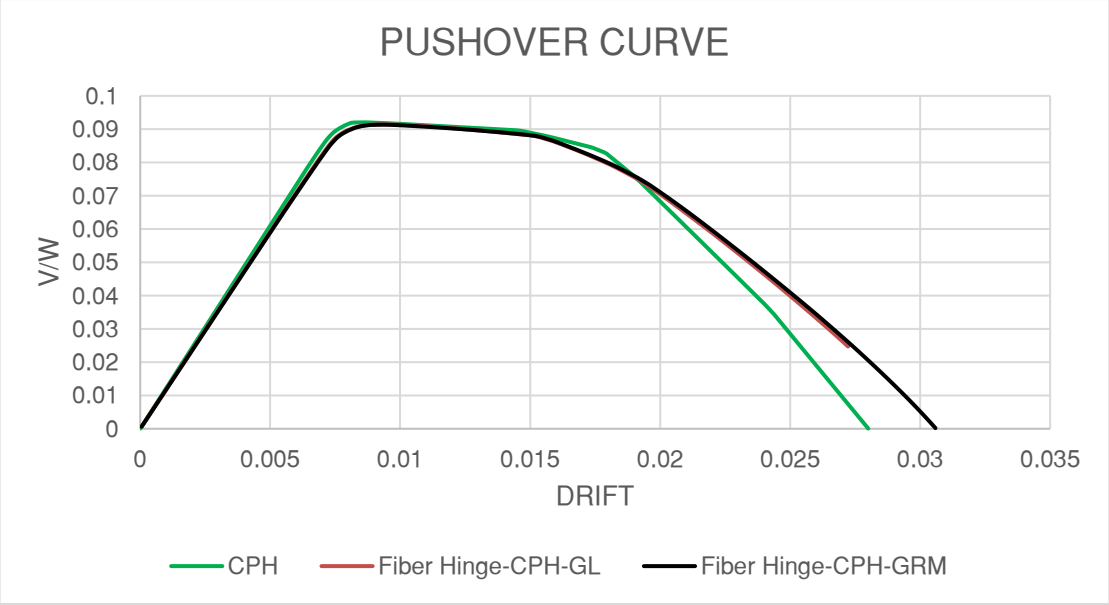
Curvas Pushover del edificio de 8 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.19

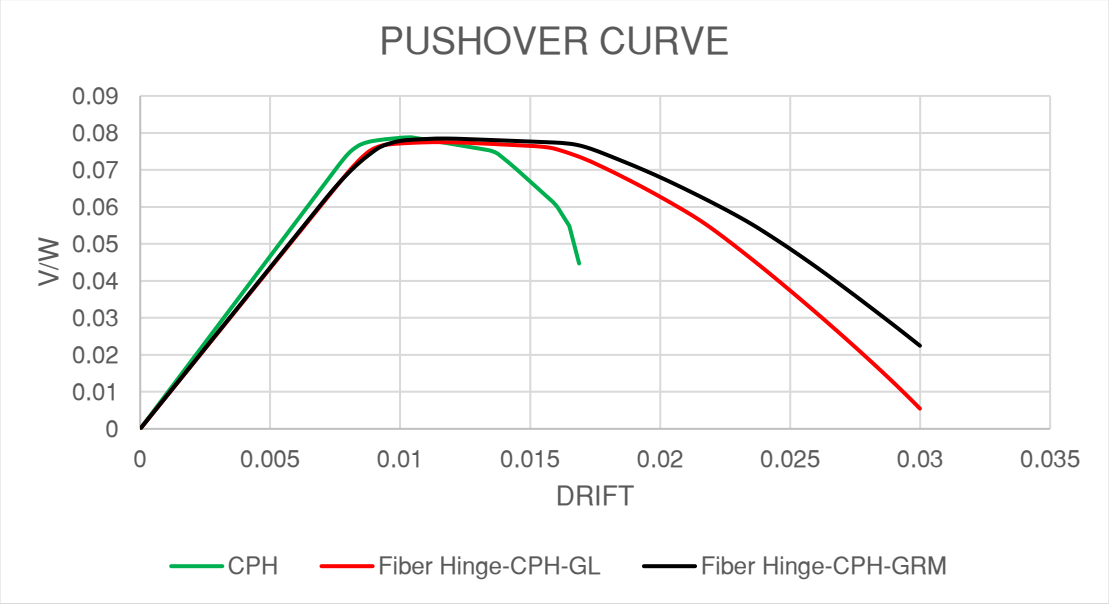
Curvas Pushover del edificio de 12 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Figura 5.20

Curvas Pushover del edificio de 20 pisos



Nota. La curva CPH corresponde a “Concentrated Plastic Hinge”.

Como se logra apreciar la convergencia es satisfactoria, y muy similar a los primeros modelos de Steel02. De igual manera se verifica un comportamiento prácticamente elástico en la estructura de 2 pisos, y en la estructura de 20 pisos, los modelos no llegan a ajustarse completamente.

Tabla 10

Propiedades de los modelos con material Steel02

# Pisos	Modelo	Integración	T (s)	ΔX (in)	X (% de H)
2	Fiber Hinge and CPH	Gauss Lobatto	0.645	0.005	60
		Gauss Radau modificado	0.653	0.005	60
4	Fiber Hinge and CPH	Gauss Lobatto	1.628	0.005	10
		Gauss Radau modificado	1.629	0.005	10
8	Fiber Hinge and CPH	Gauss Lobatto	2.248	0.01	10
		Gauss Radau modificado	2.261	0.01	10
12	Fiber Hinge and CPH	Gauss Lobatto	3.024	0.1	10
		Gauss Radau modificado	3.033	0.1	10
20	Fiber Hinge and CPH	Gauss Lobatto	4.318	0.1	3
		Gauss Radau modificado	4.296	0.1	3

Nota. "H" es la altura total del edificio correspondiente

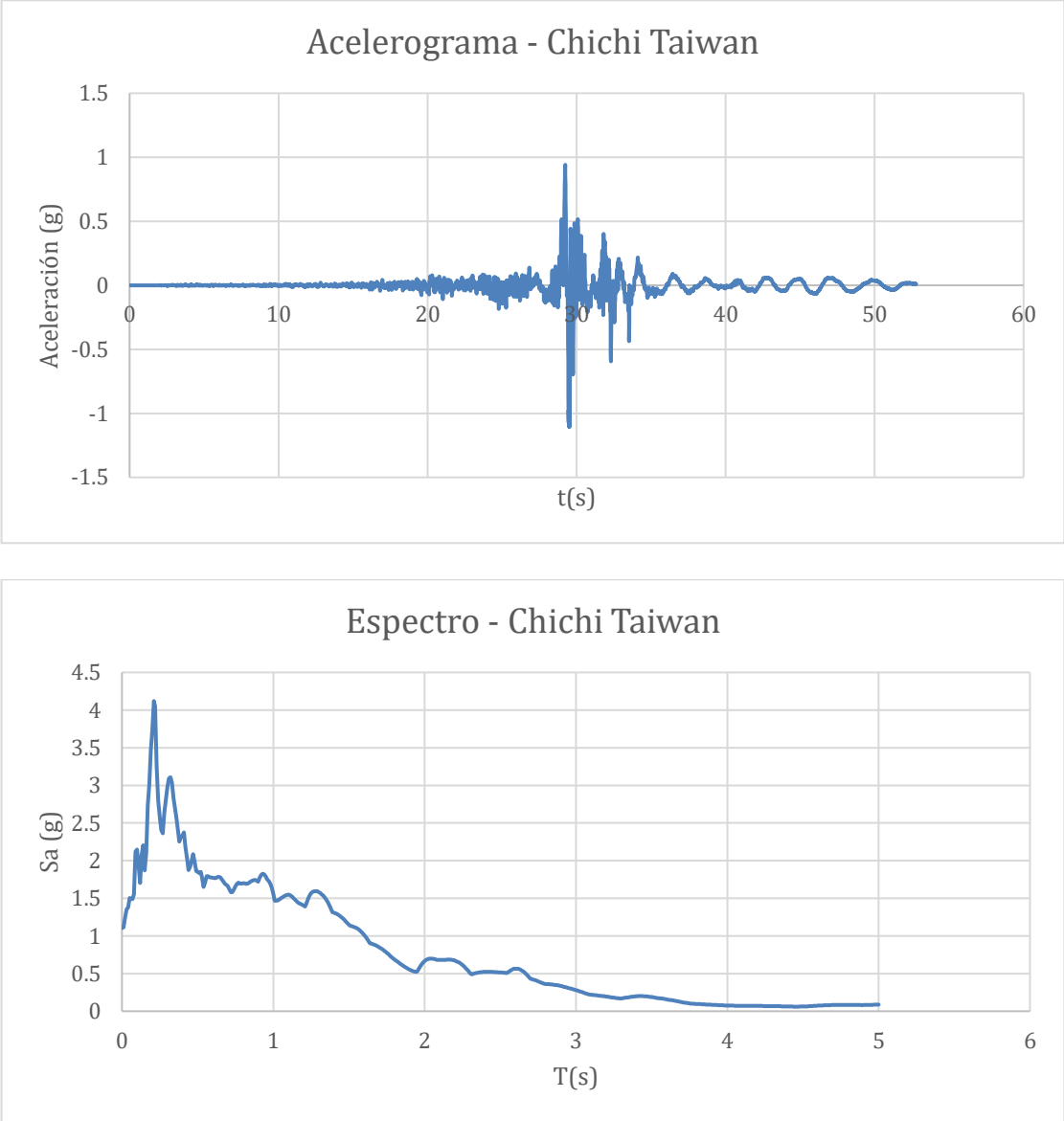
De igual manera se logra apreciar que el costo computacional es medianamente significativo, siendo a los que más les cuesta converger a los modelos de 2 y 4 pisos. Mientras que, en los modelos de 12 y 20 pisos, su convergencia es más rápida.

5.2. Análisis Dinámico

Se decidió graficar el acelerograma y el espectro de aceleraciones, para tener una idea de a que tipo de estructuras podría afectar el registro aplicado, y si es congruente con los resultados obtenidos. De igual manera se graficó las ubicaciones de los puntos de integración, para entender mejor el comportamiento de los resultados.

Figura 5.21

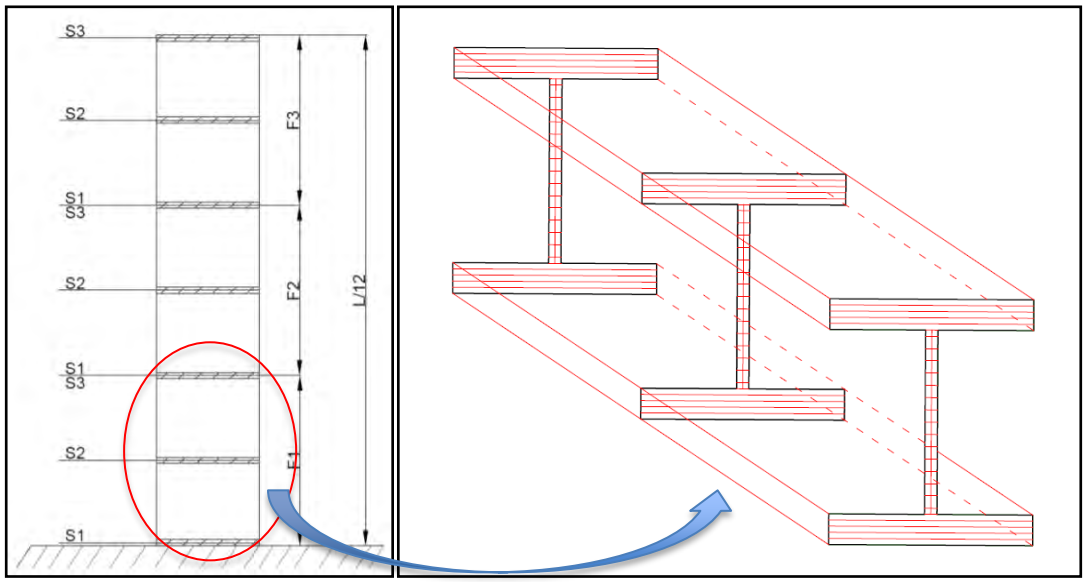
Acelerograma y Espectro del registro alpicado



Nota. El PGA es 1.11 g. La duración del evento es de 52.79 segundos.

Figura 5.22

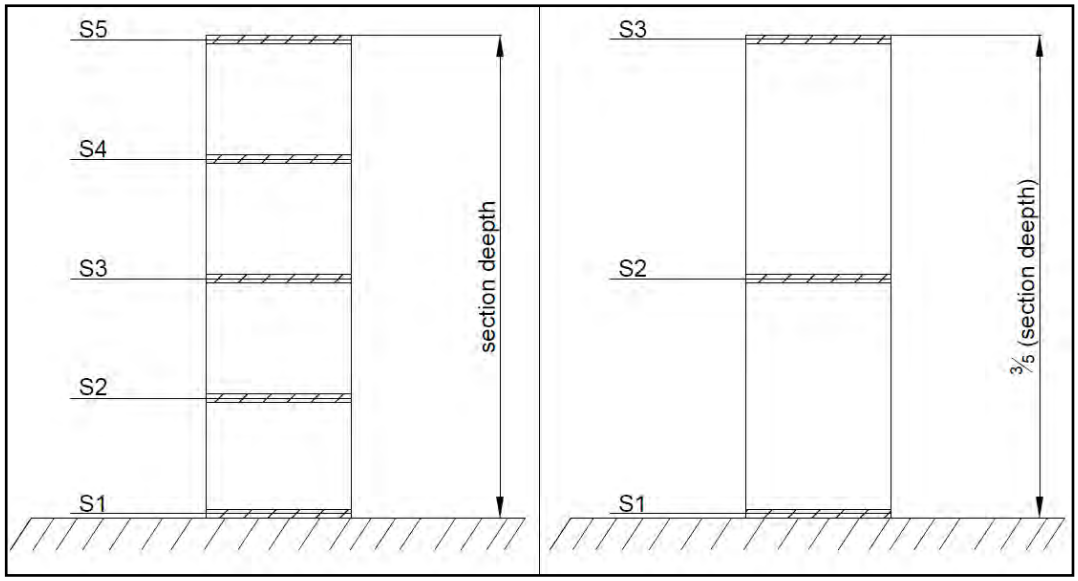
Esquema y nomenclatura de la disposición de fibras de los modelos L/12



Nota. "S" corresponde a la ubicación del punto de integración de la fibra en cuestión.

Figura 5.23

Esquema y nomenclatura de la disposición de fibras de los modelos section deepth



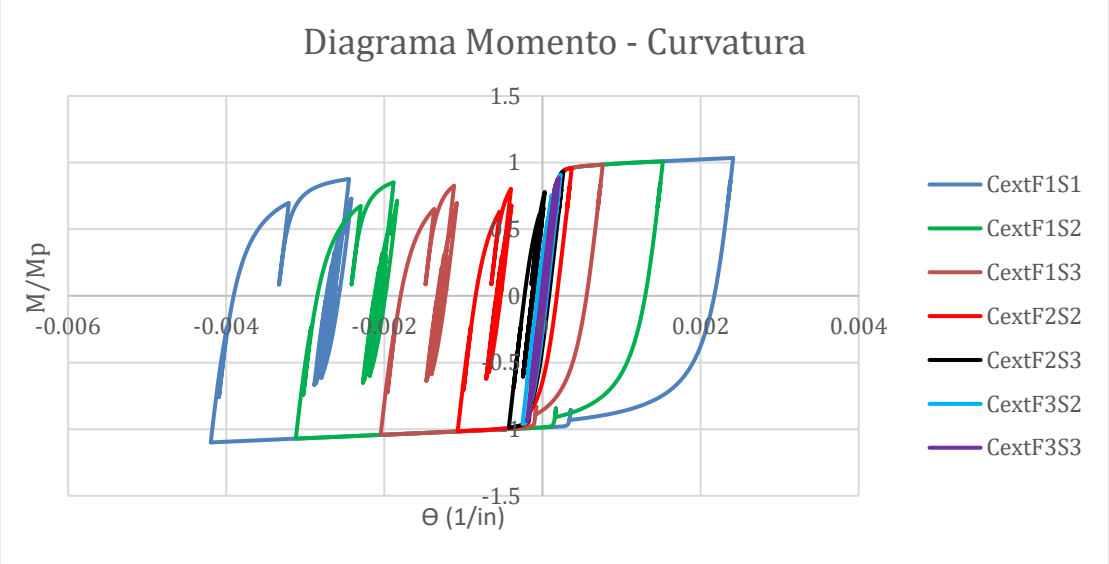
Nota. La figura de la izquierda corresponde a los modelos "GL" y la de la derecha a los modelos "GRM"

5.2.1. Modelos con Longitud de Rótula Plástica de L/12 y Material Steel02

Para obtener el momento probable (Mp), se utilizó $Mp = 1.1 * Fy * Zx$, donde Zx es el módulo plástico de la sección. De igual manera es necesario recalcar el hecho de que los resultados de Opensees son en el orden de "Axial, Momento" en el caso de fuerzas y "Axial, curvatura" en el caso de deformaciones.

Figura 5.24

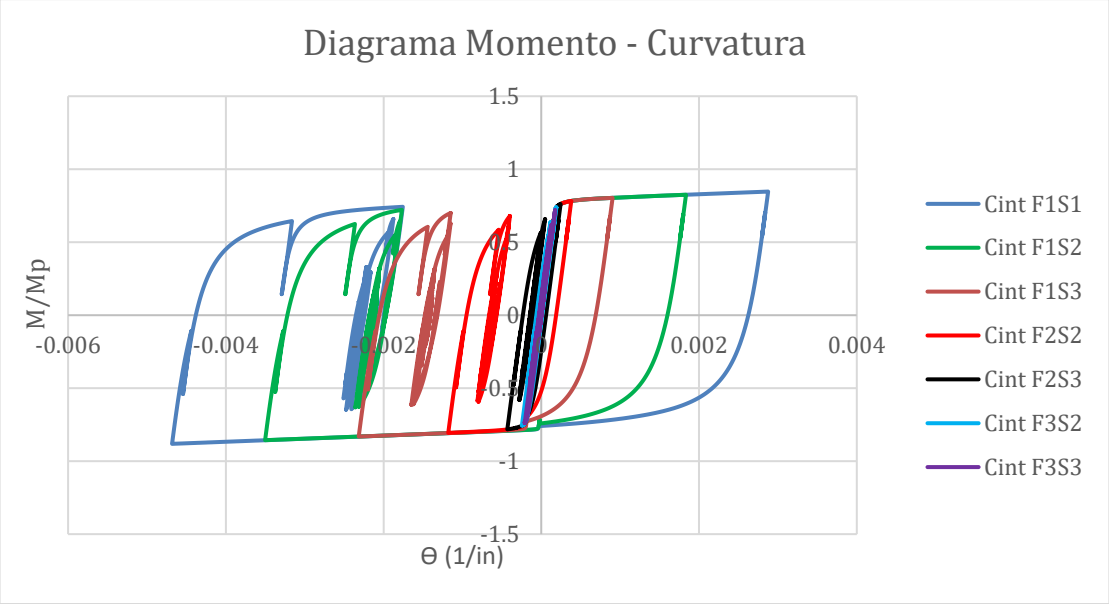
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.25

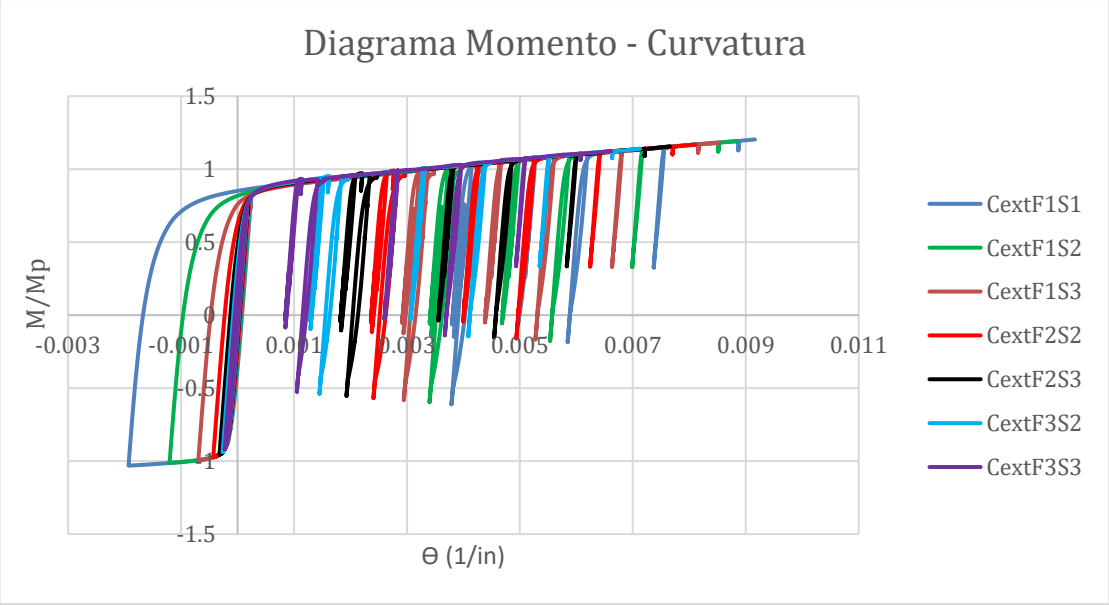
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.26

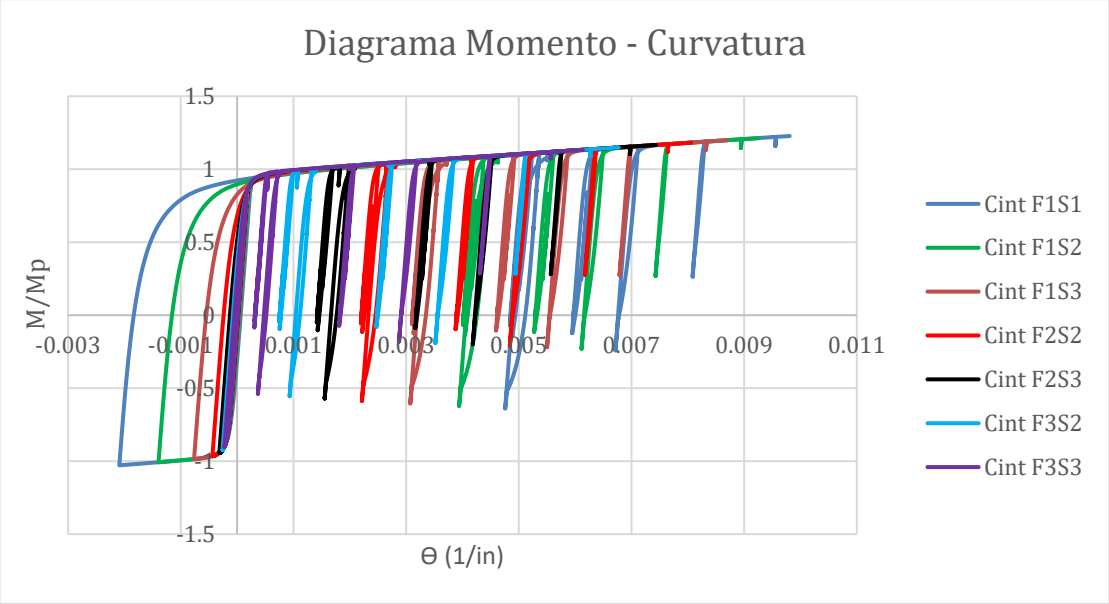
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.27

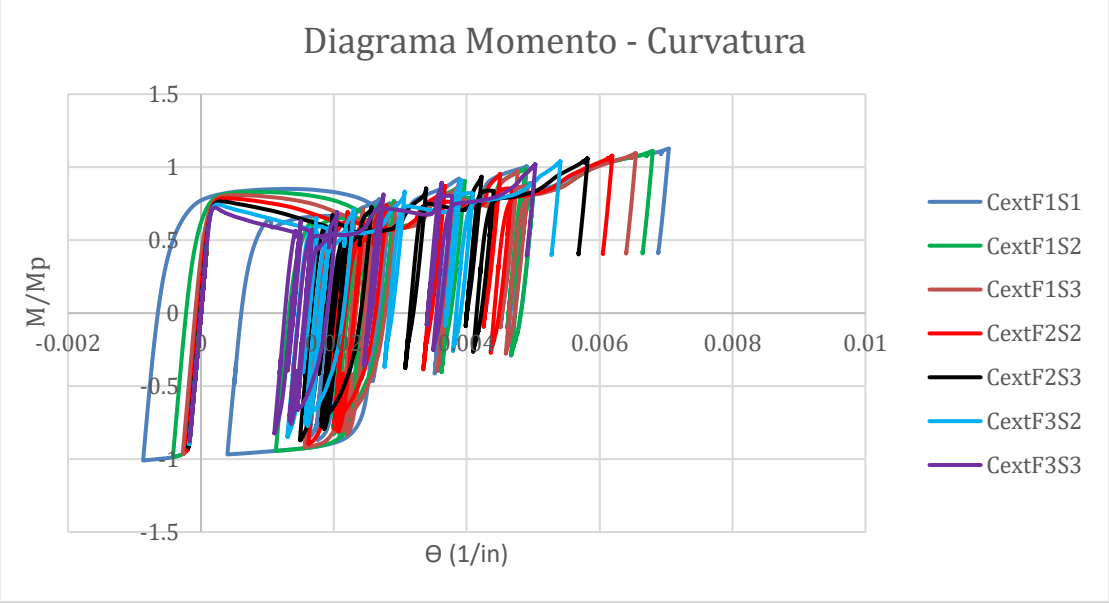
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.28

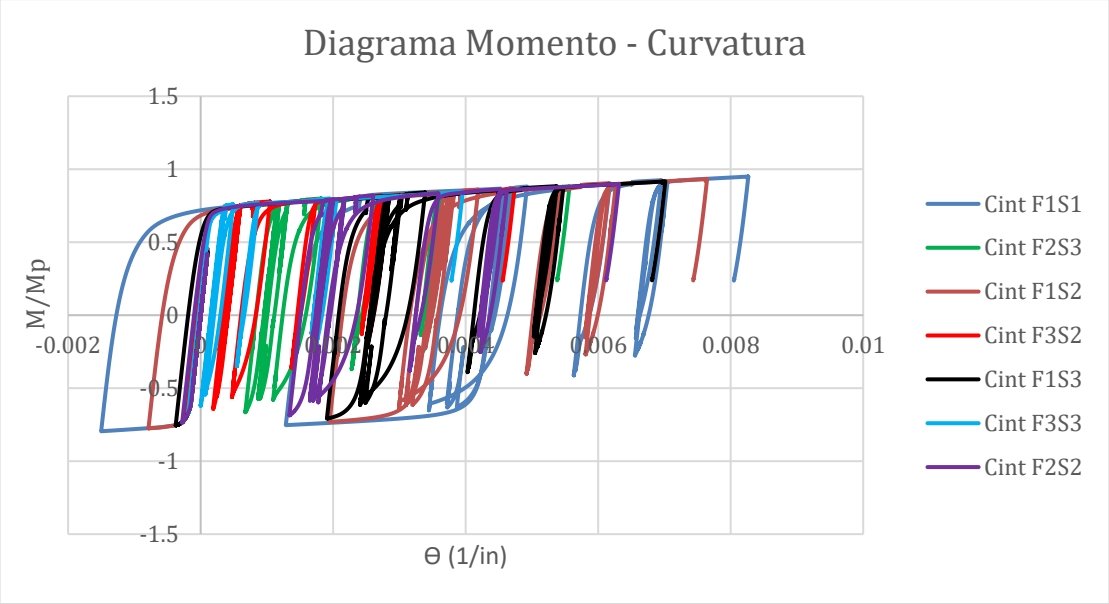
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.29

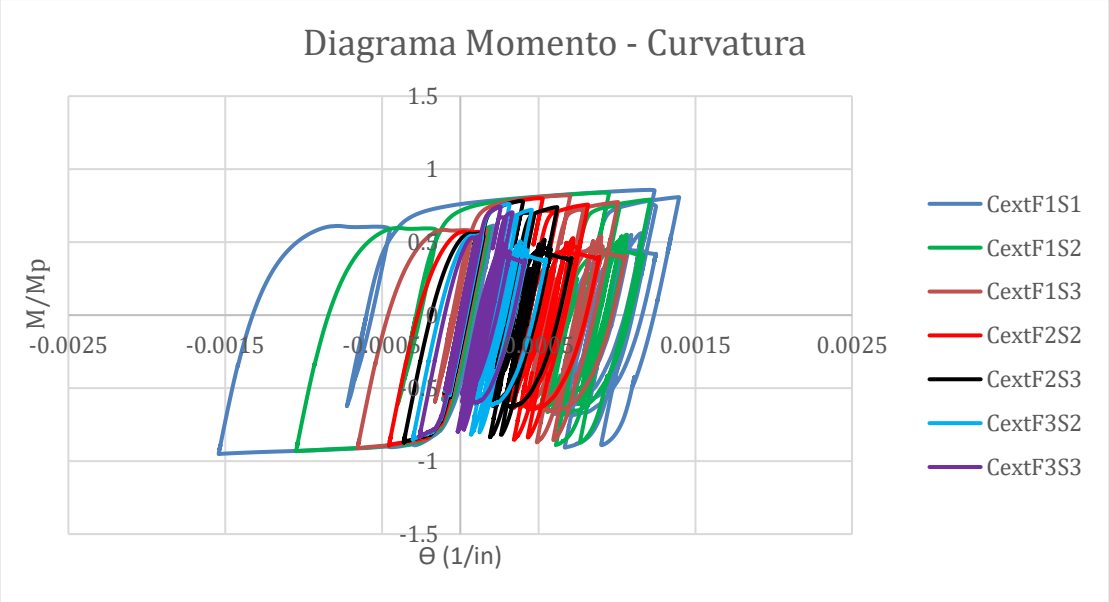
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.30

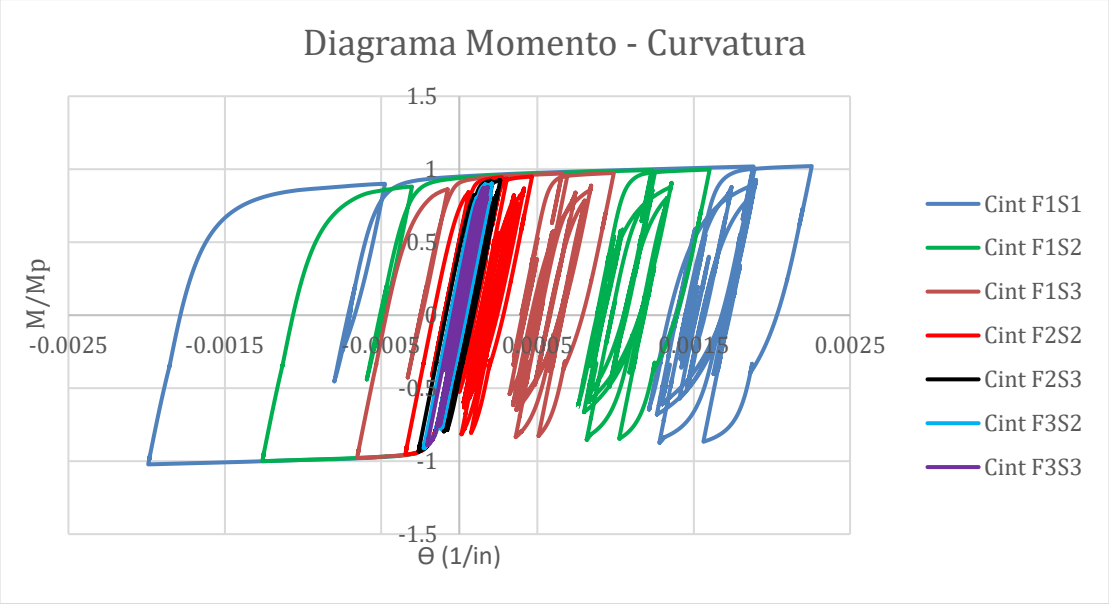
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.31

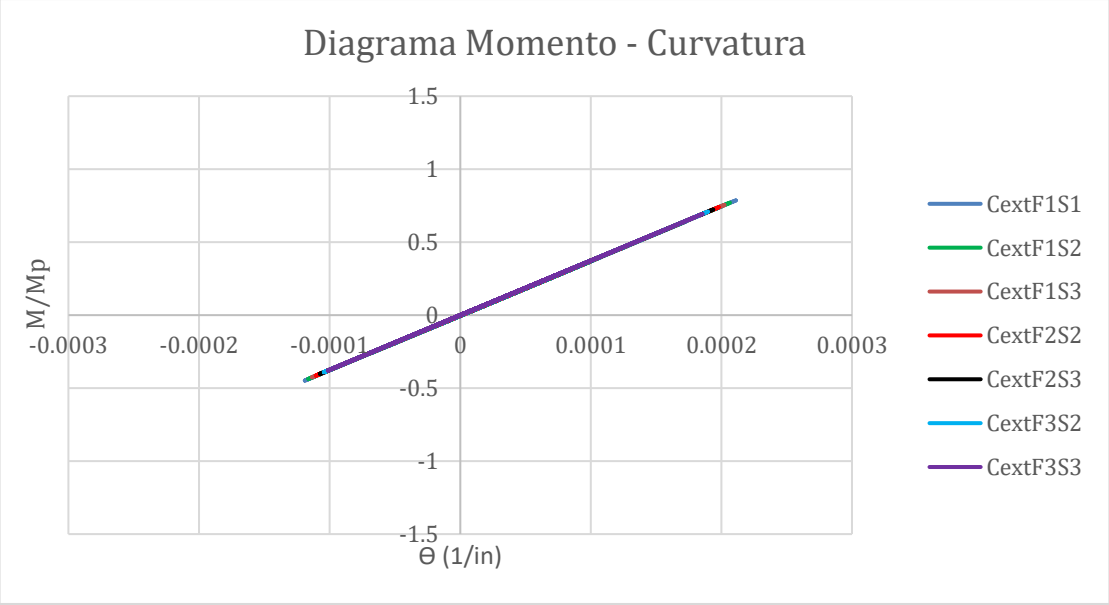
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.32

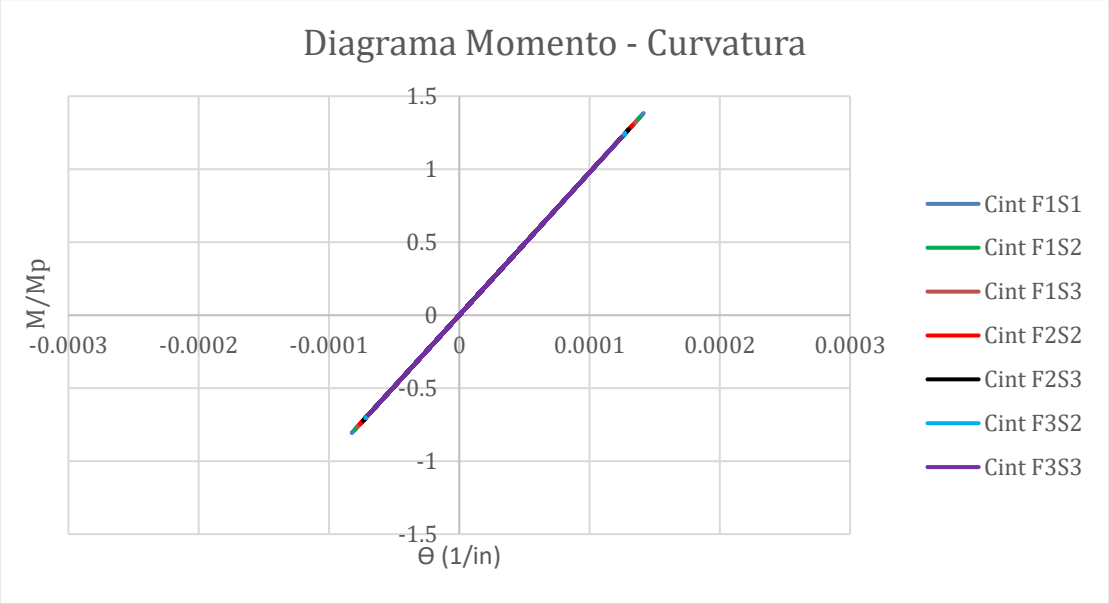
Diagrama momento-curvatura de la estructura de 20 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.33

Diagrama momento-curvatura de la estructura de 20 pisos



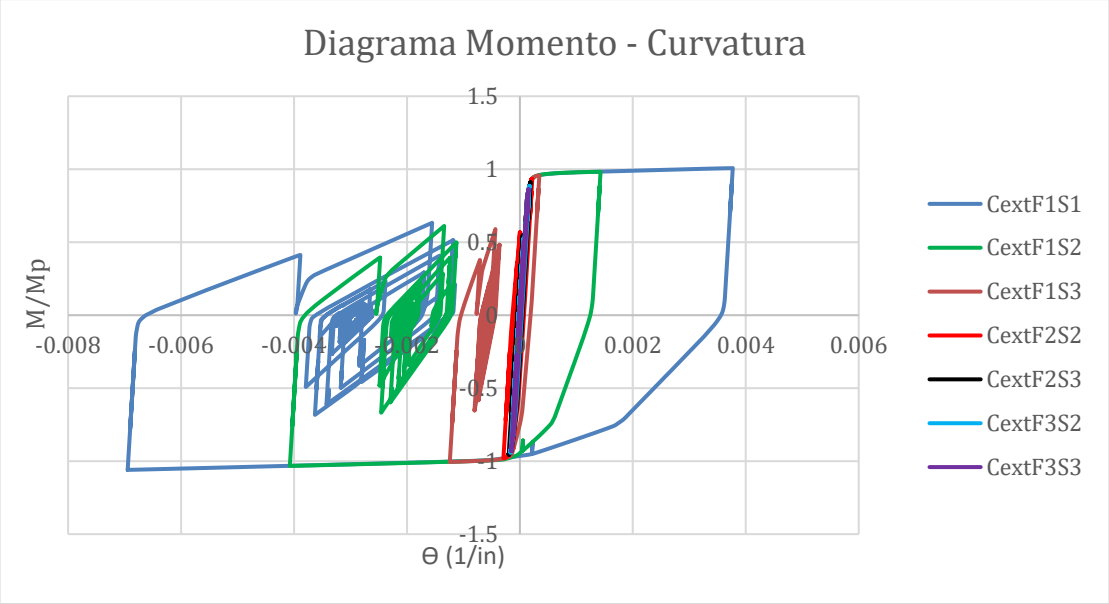
Nota. El diagrama pertenece a la columna interior de la estructura

5.2.2. Modelos con Longitud de Rótula Plástica de L/12 y Material Hysteretic

El material Hysteretic tuvo como objetivo el mostrar la variación con el material Steel02, desde el punto de vista de la definición. Y estimar cuales son las implicaciones entre usar uno u otro en proyectos.

Figura 5.34

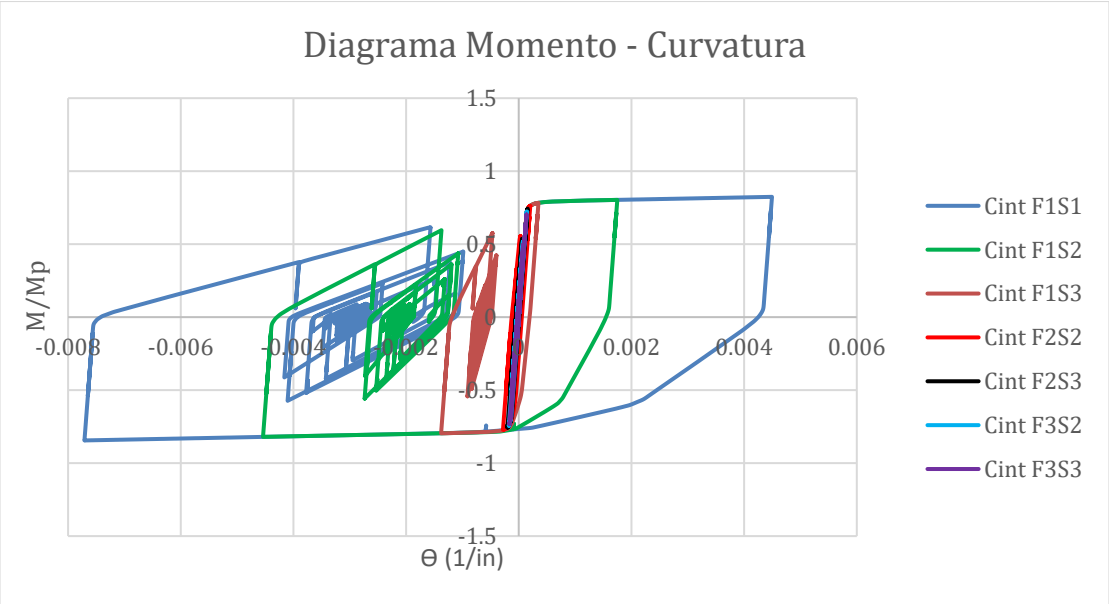
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.35

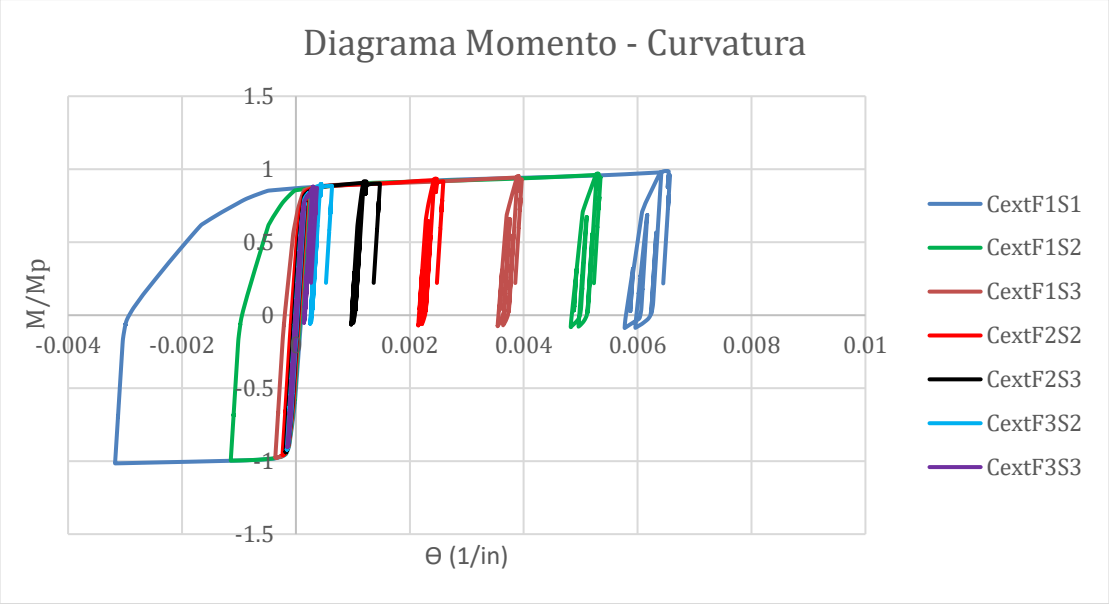
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.36

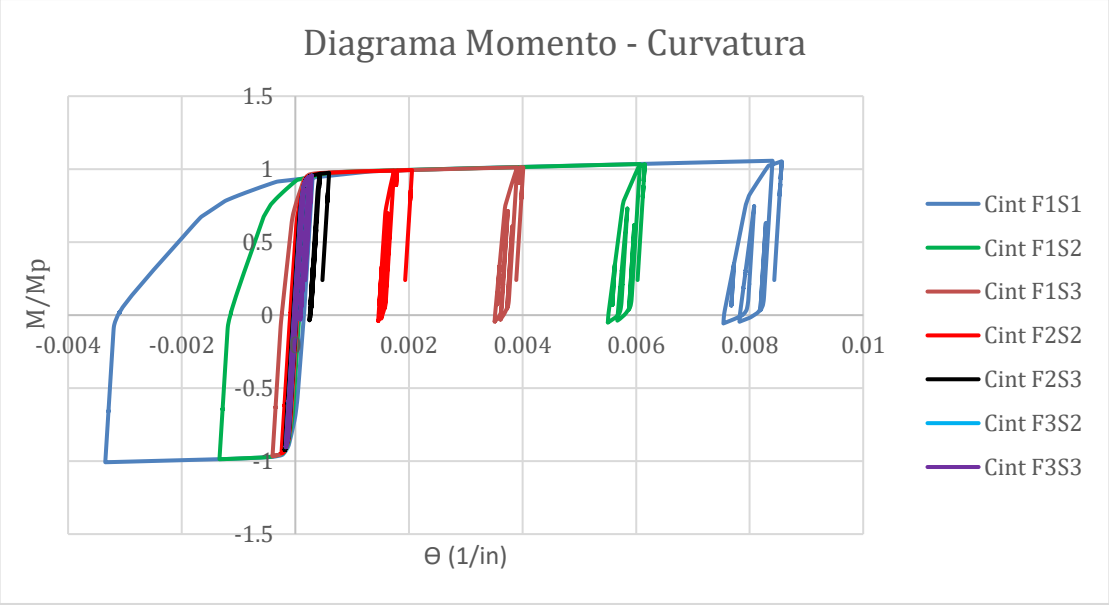
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.37

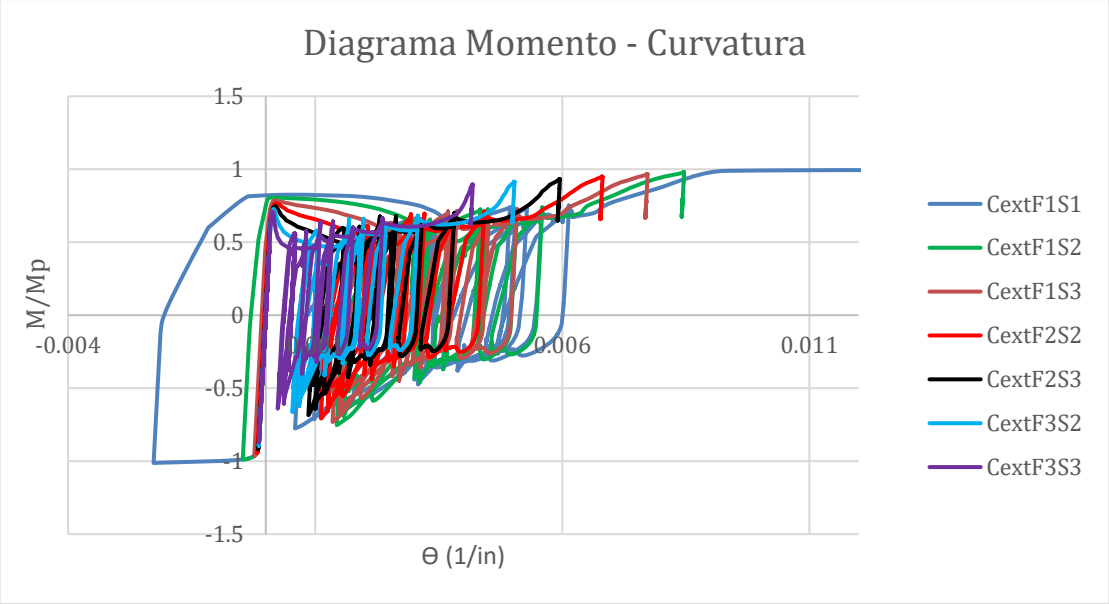
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.38

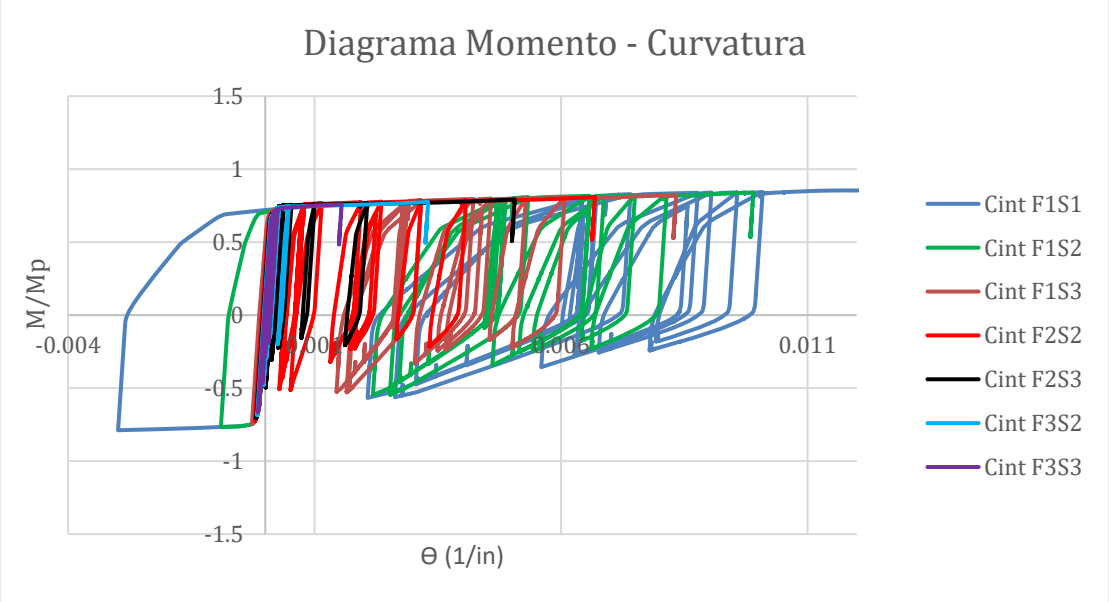
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.39

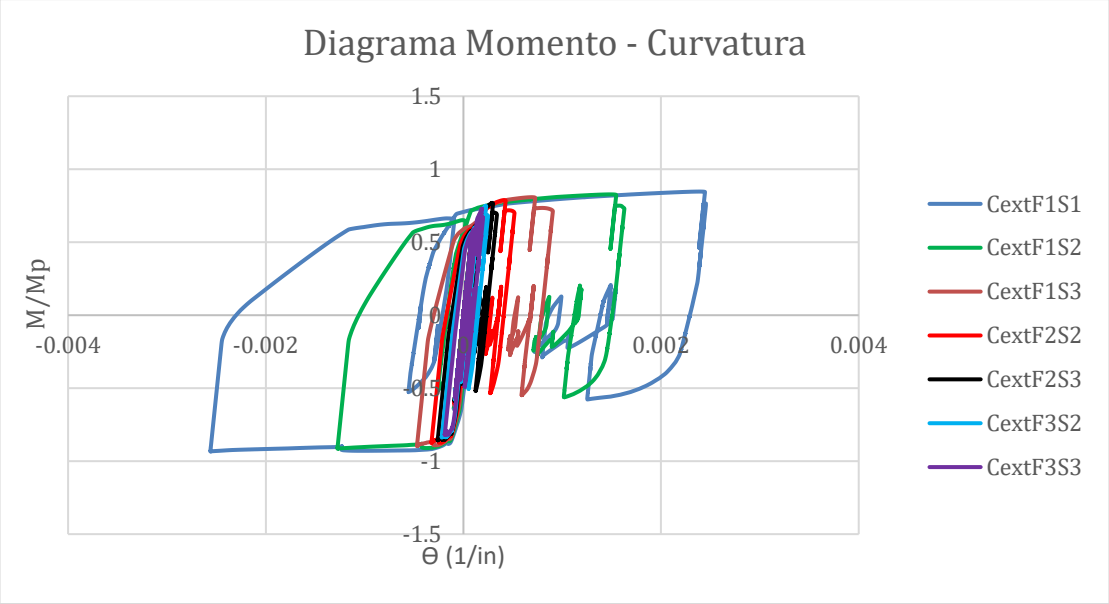
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.40

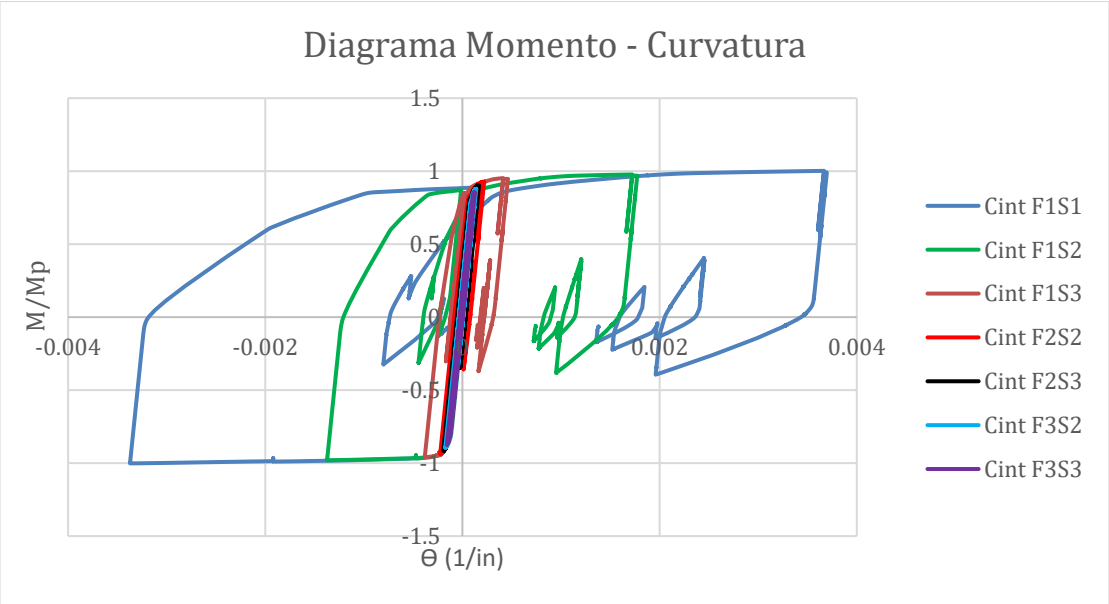
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.41

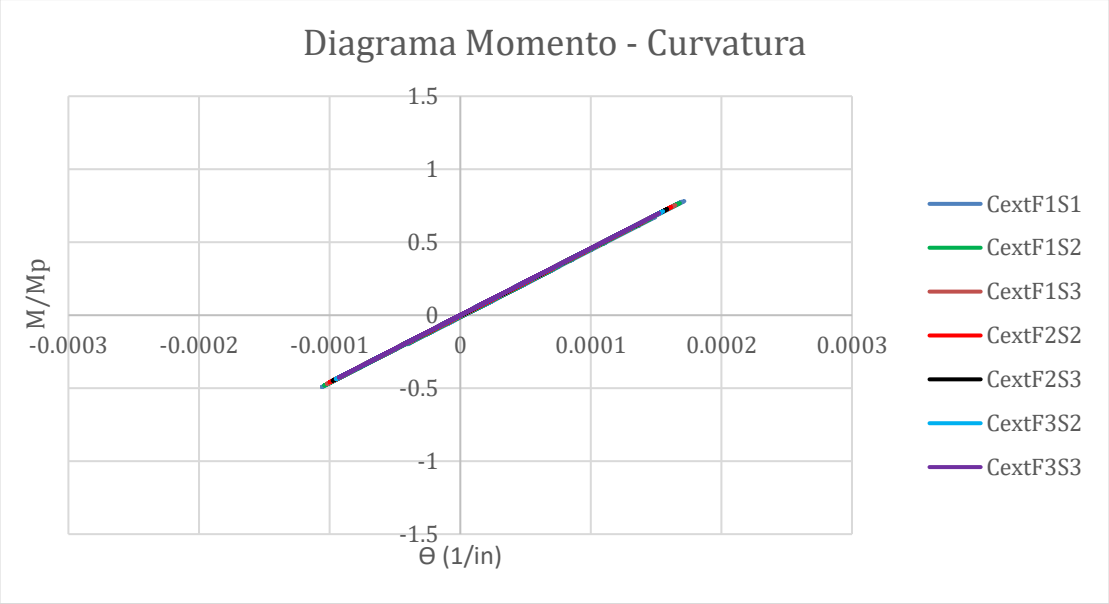
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.42

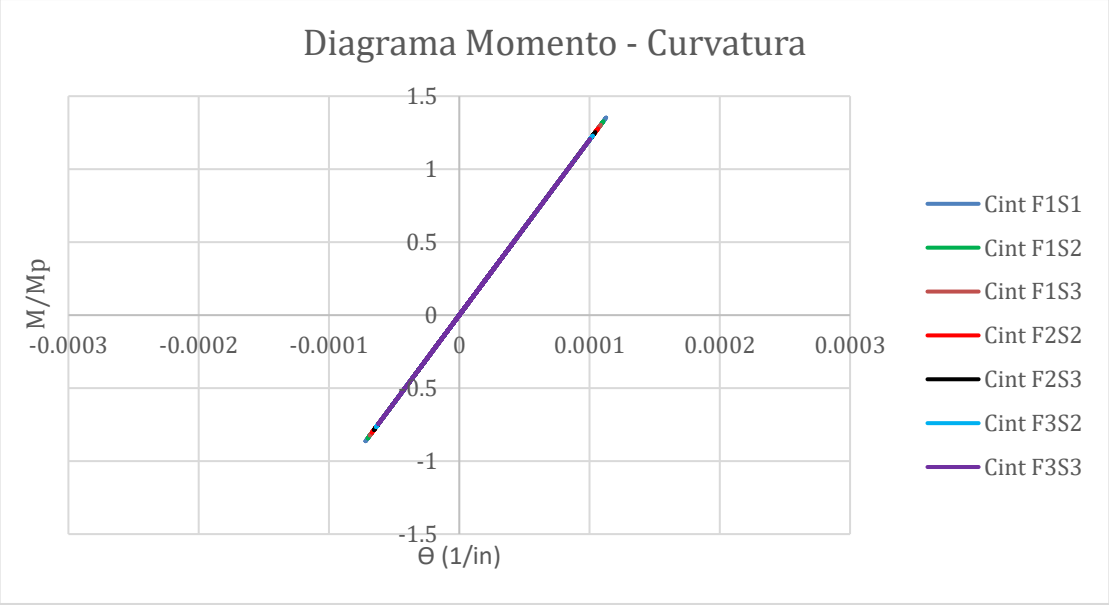
Diagrama momento-curvatura de la estructura de 20 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.43

Diagrama momento-curvatura de la estructura de 20 pisos



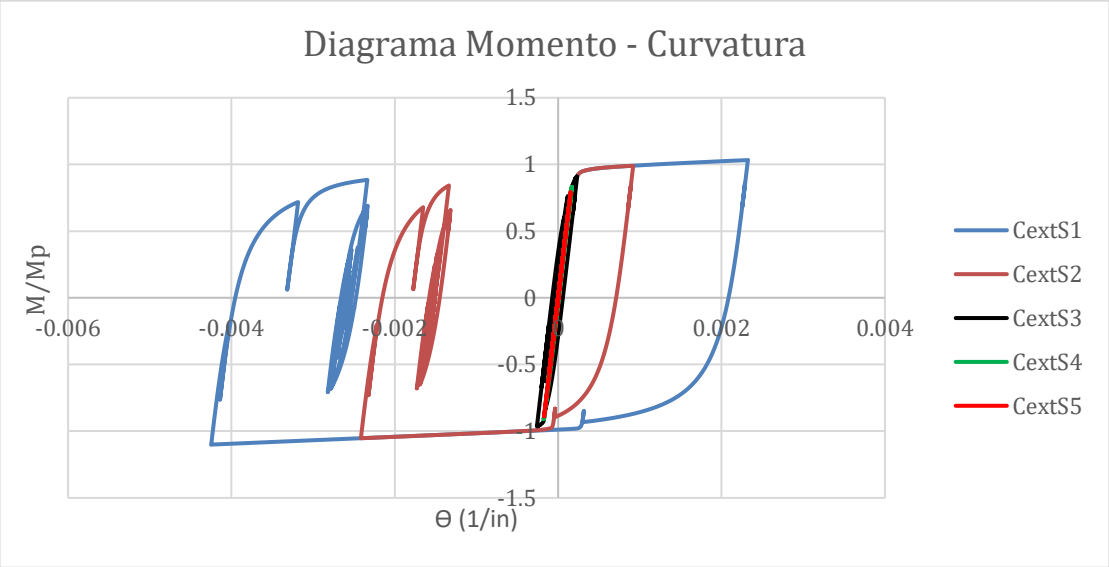
Nota. El diagrama pertenece a la columna interior de la estructura

5.2.3. Modelos con Longitud de Rótula Plástica de Section Deepth (GL)

Puesto a que ya se había hecho la comparativa entre materiales en la parte dinámica, se decide solo usar el material Steel02 para las distribuciones. Como el caso de las siguientes gráficas, en las cuales se grafican el aporte de todos los puntos de integración.

Figura 5.44

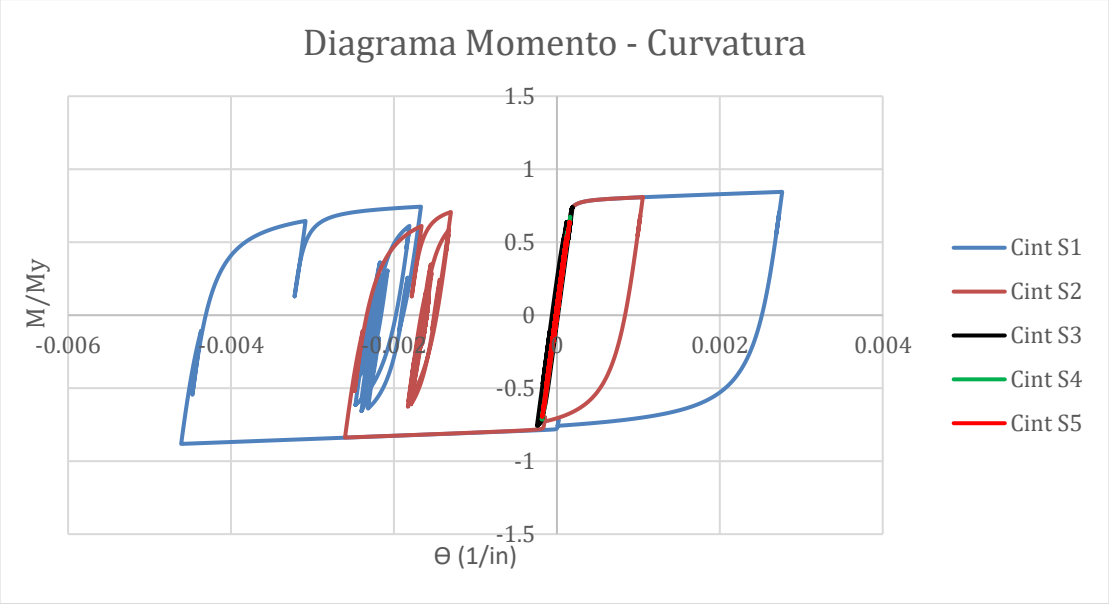
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.45

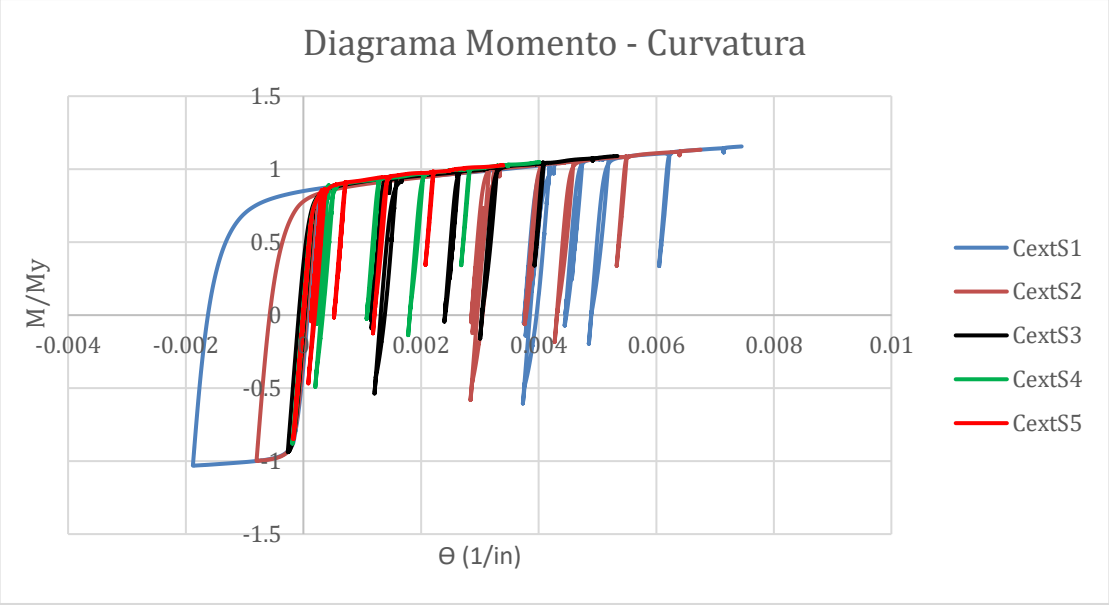
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.46

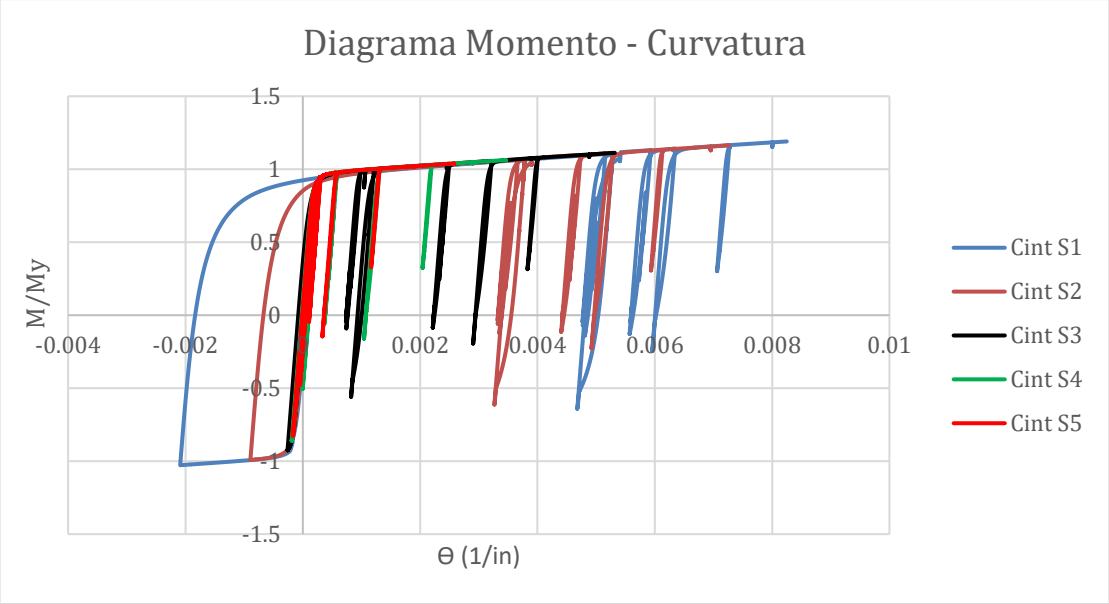
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.47

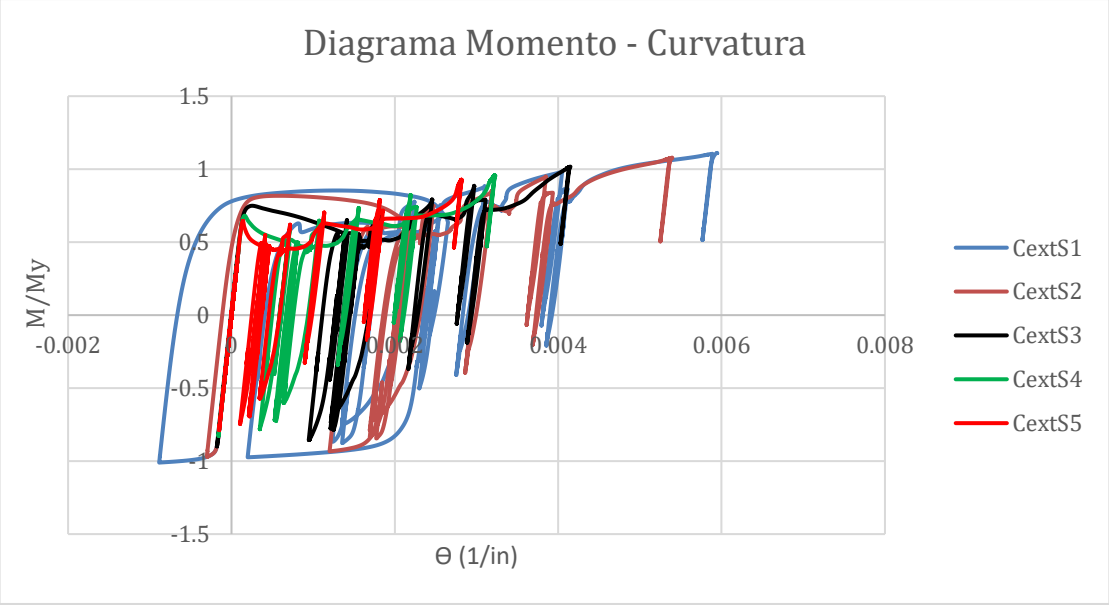
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.48

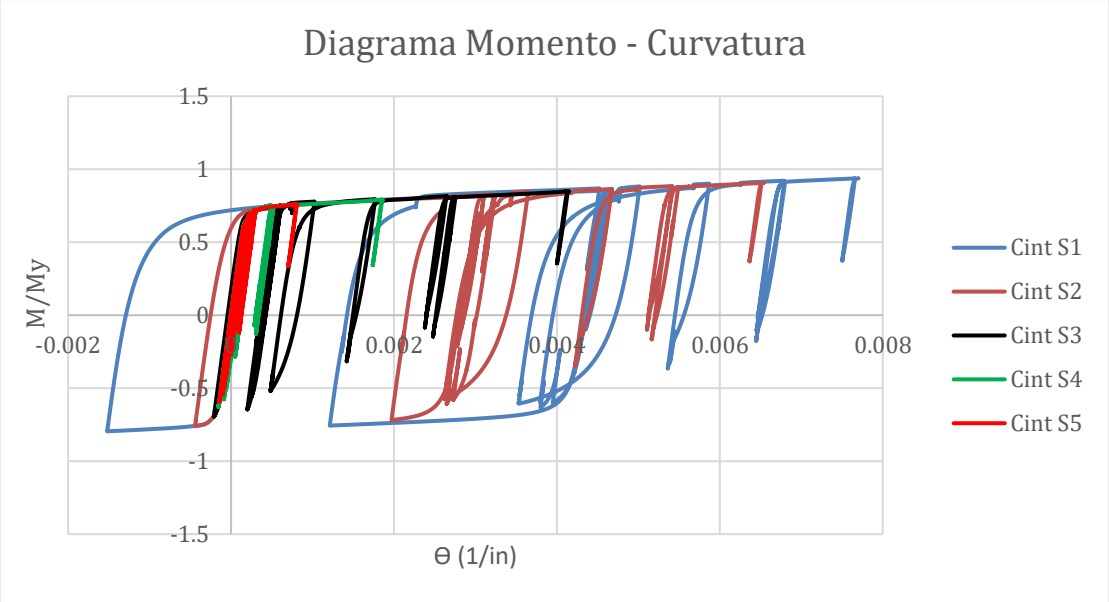
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.49

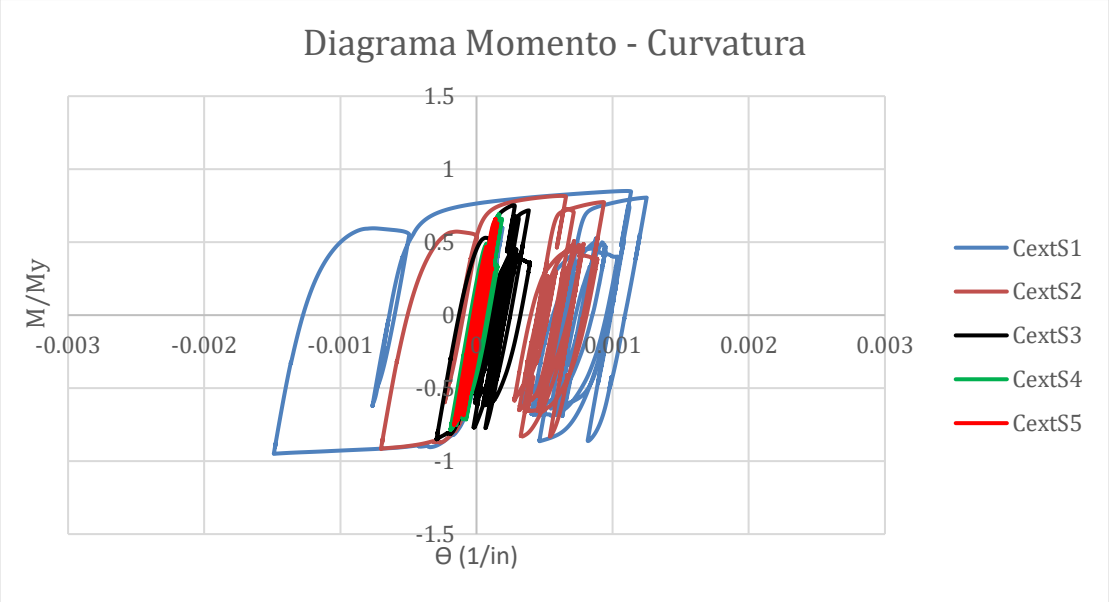
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.50

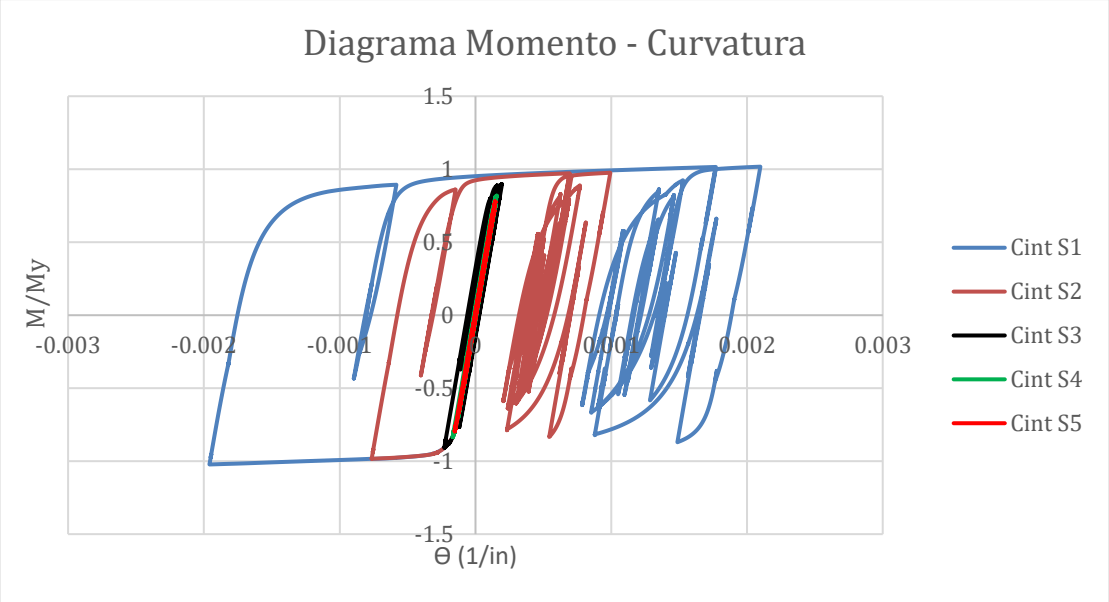
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.51

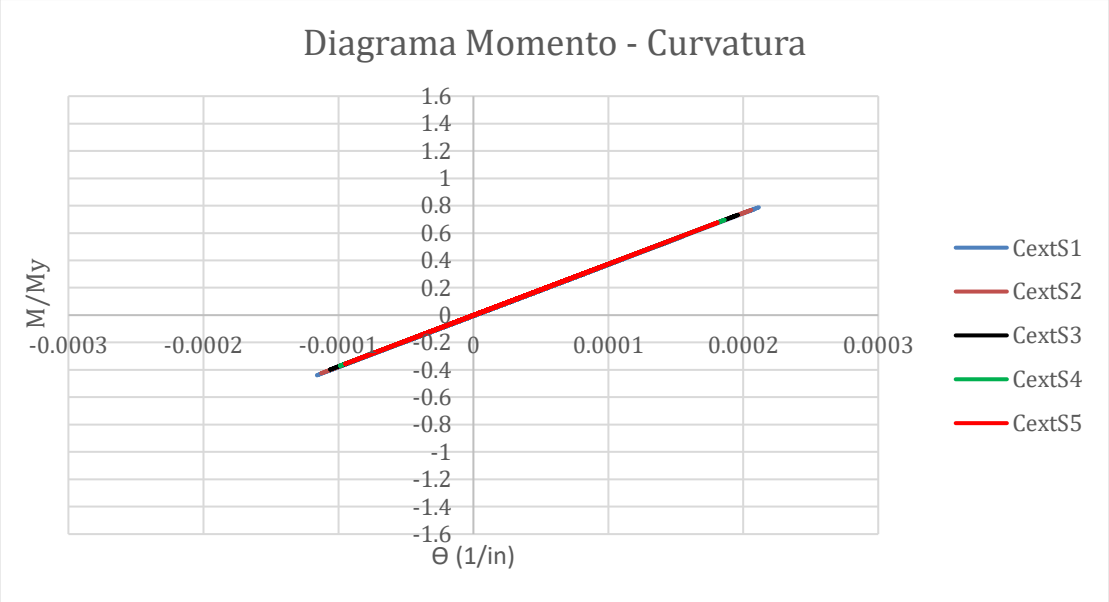
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.52

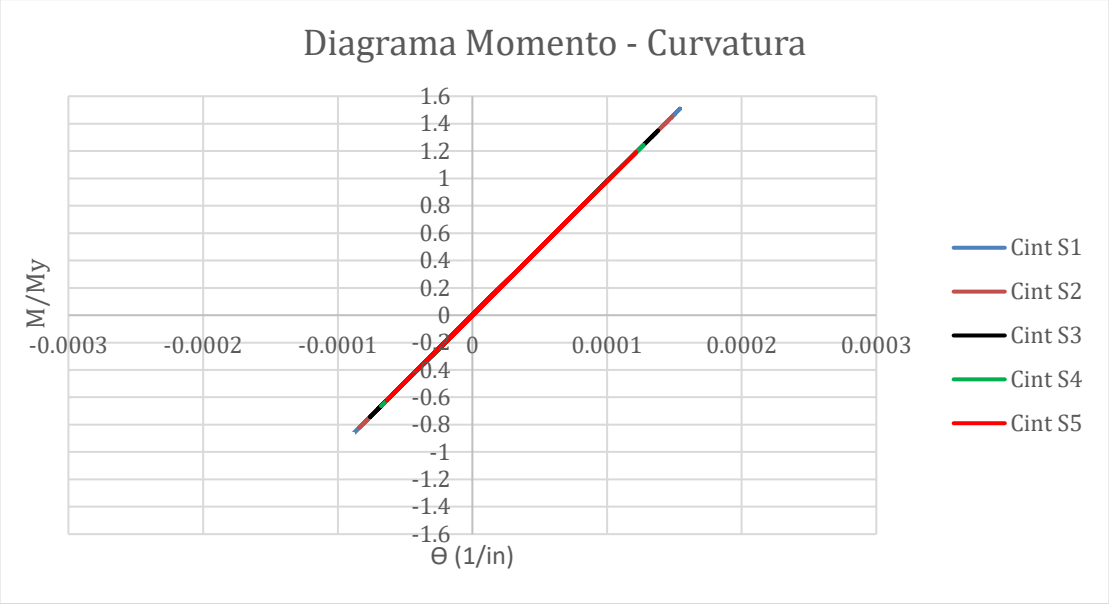
Diagrama momento-curvatura de la estructura de 20 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.53

Diagrama momento-curvatura de la estructura de 20 pisos



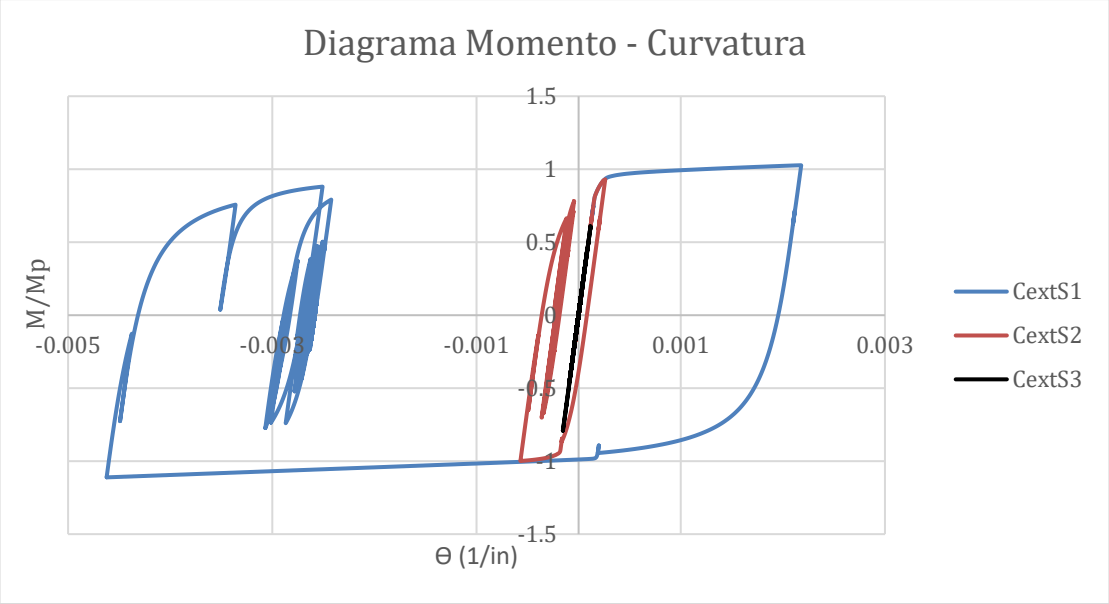
Nota. El diagrama pertenece a la columna interior de la estructura

5.2.4. Modelos con Longitud de Rótula Plástica de 3/5(d) (GRM)

Para las gráficas a continuación también se usó material Steel02, y se estimó que la cantidad de fibras usadas por el programa fueron de 3. Puesto que la tendencia hasta el momento siempre era ir de mayor a menor. Siendo la mayor siempre la sección 1.

Figura 5.54

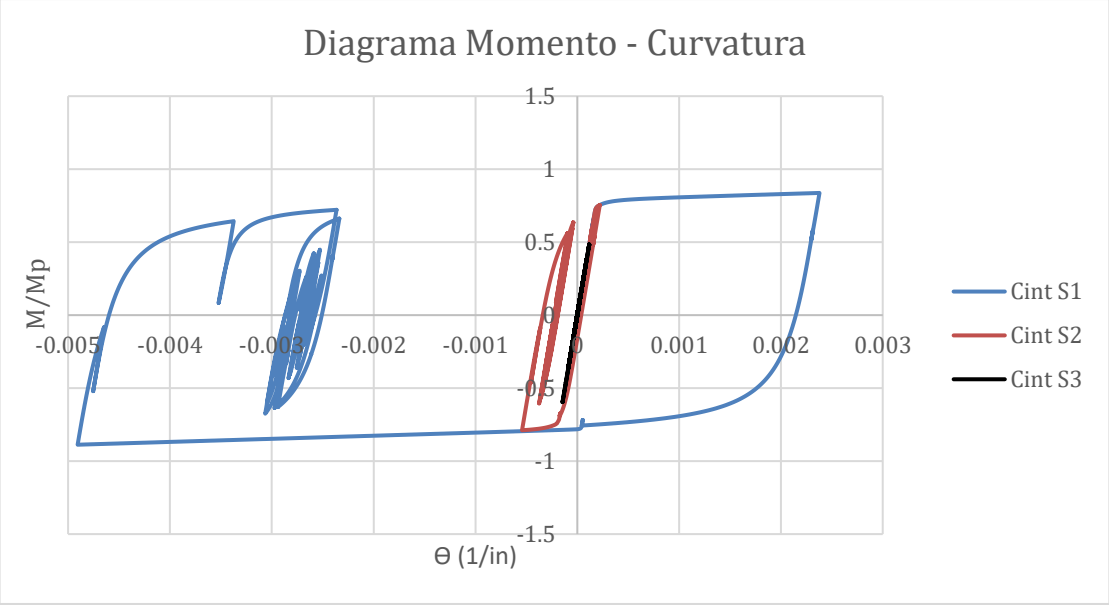
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.55

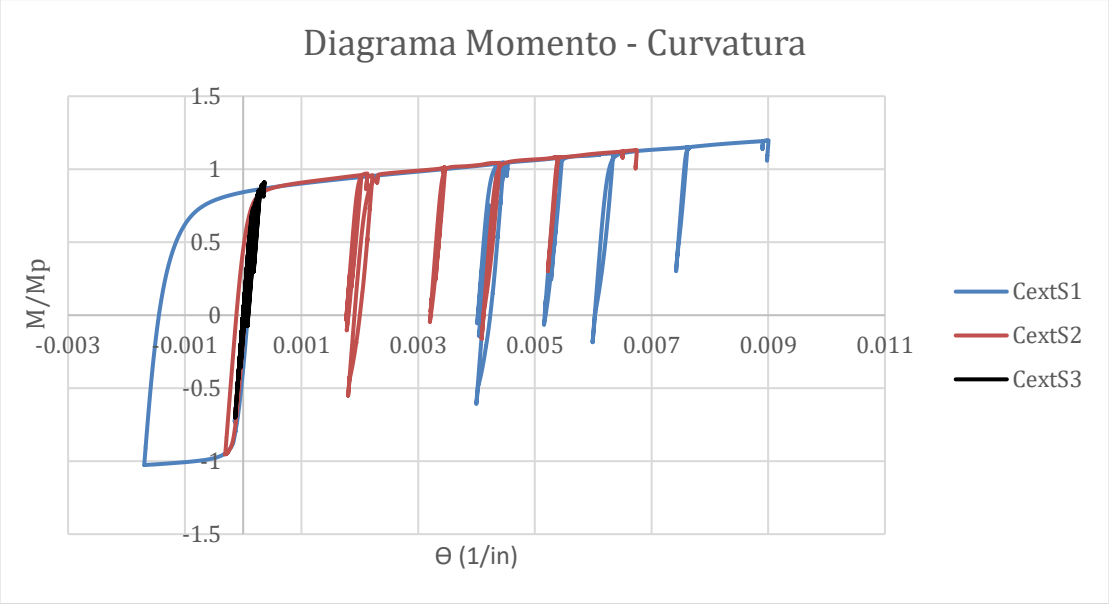
Diagrama momento-curvatura de la estructura de 2 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.56

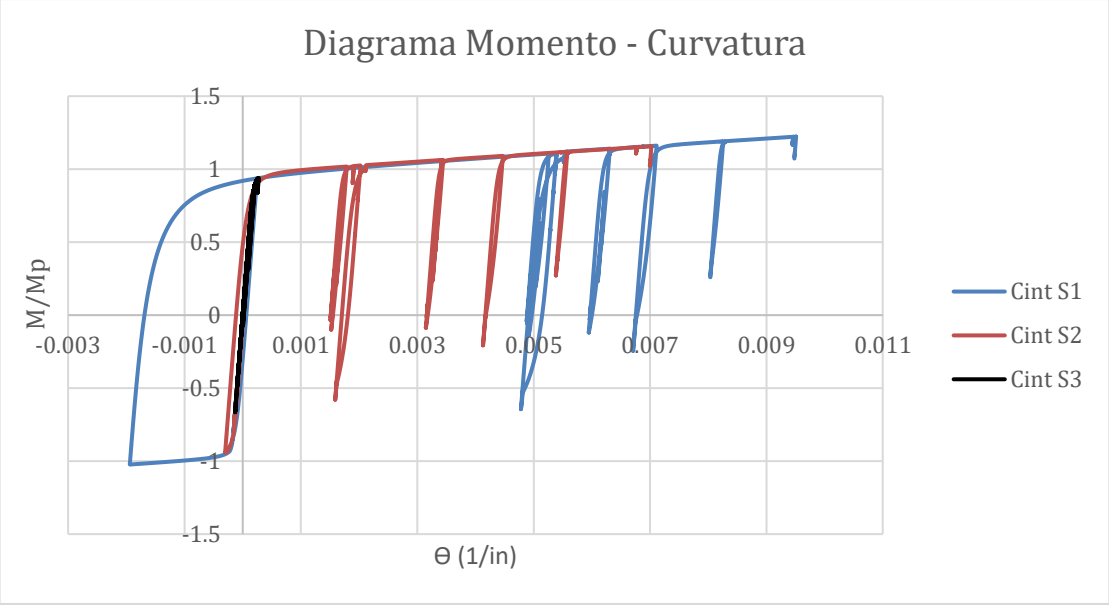
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.57

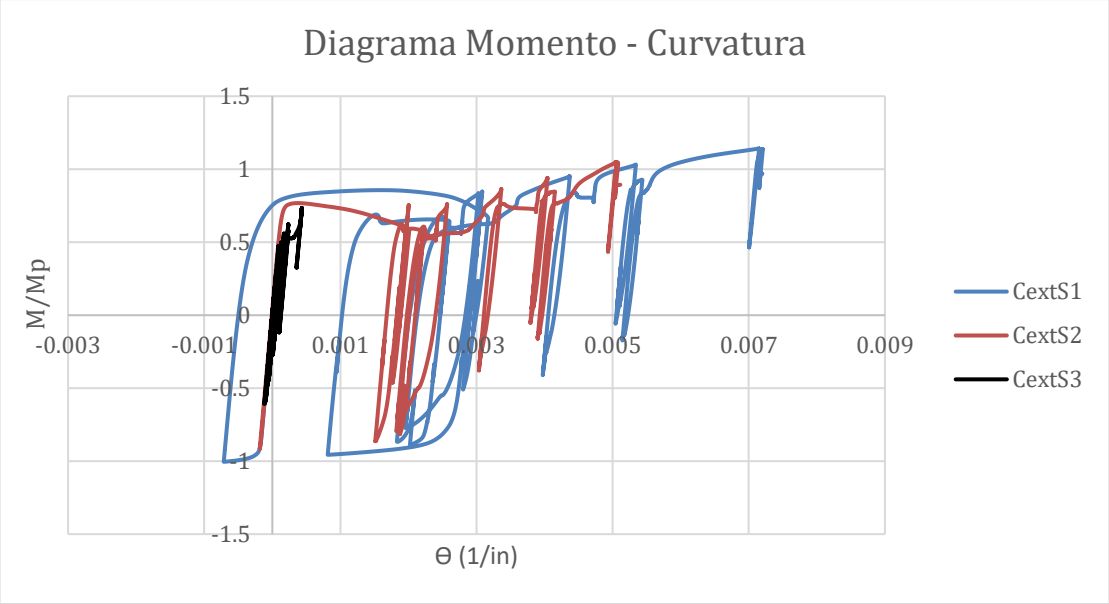
Diagrama momento-curvatura de la estructura de 4 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.58

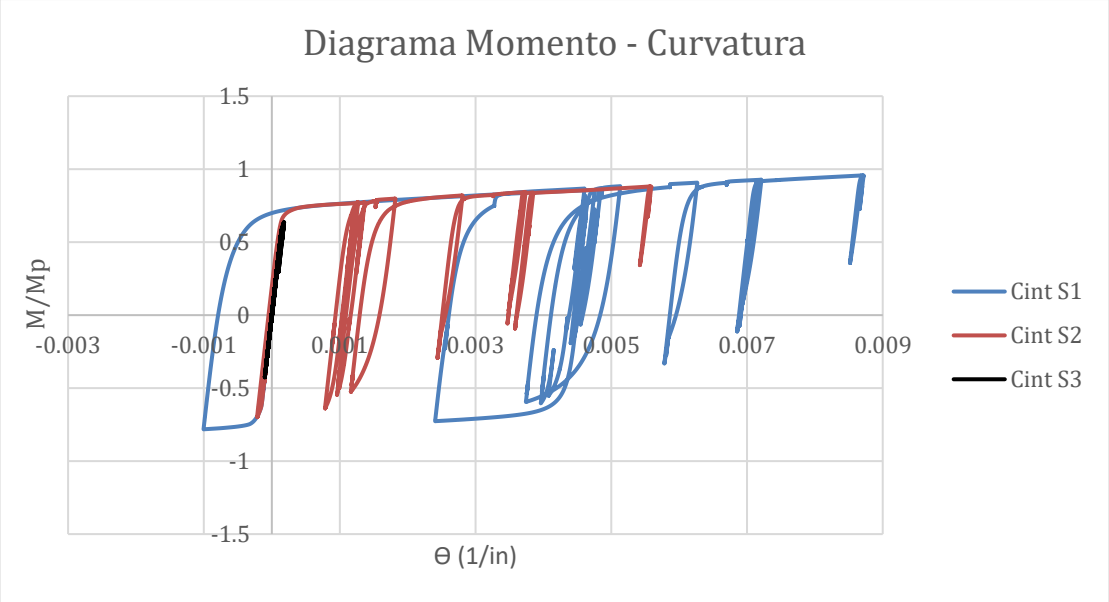
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.59

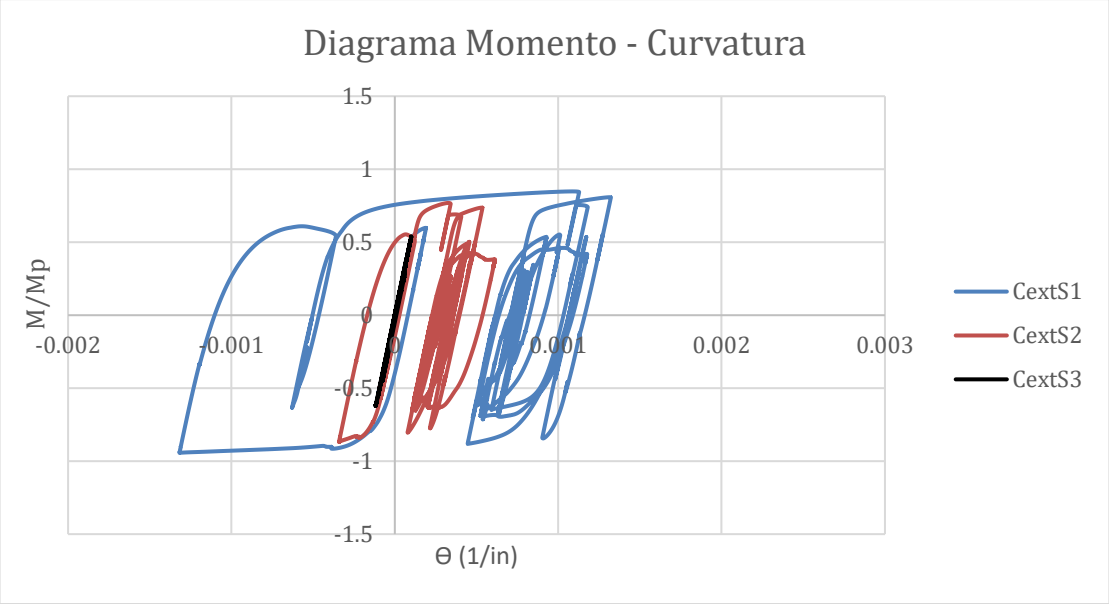
Diagrama momento-curvatura de la estructura de 8 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.60

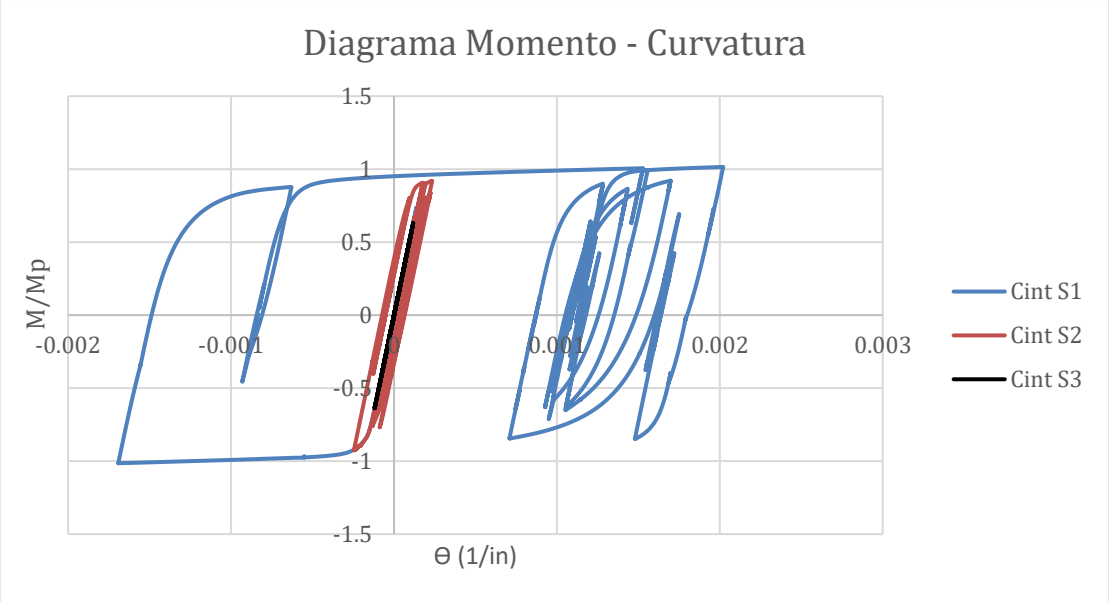
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.61

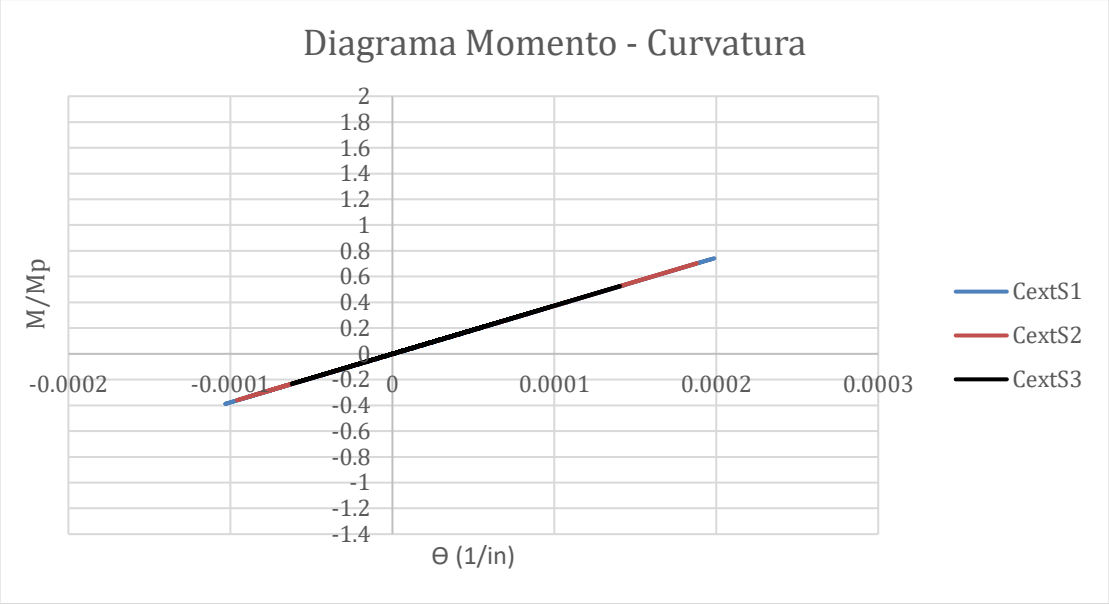
Diagrama momento-curvatura de la estructura de 12 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Figura 5.62

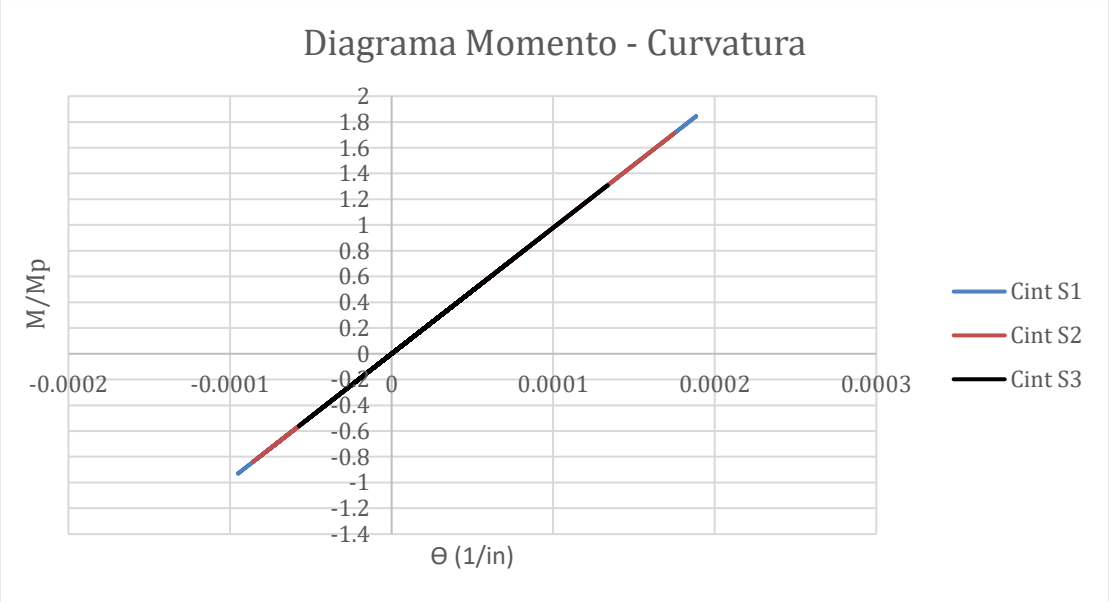
Diagrama momento-curvatura de la estructura de 20 pisos



Nota. El diagrama pertenece a la columna exterior de la estructura

Figura 5.63

Diagrama momento-curvatura de la estructura de 20 pisos



Nota. El diagrama pertenece a la columna interior de la estructura

Se puede observar que los diagramas de la estructura de 2 pisos entre steel02 e Hysteretic a simple viste se comportan de manera similar, tanto para la columna exterior como para la columna interior. Puesto que en ambos la columna exterior llega a la fluencia en un valor muy cercano al Mp (Momento probable). De igual manera en ambos la columna interior llega a fluir en un valor cercano al 80% del Mp. Sin embargo, se evidencia que con el material Hysteretic la capacidad de deformación de la sección es mayor, en columna exterior e interior.

En la estructura de 4 pisos se presenta colapso para ambos materiales. Debido a esto la convergencia de los diagramas se ve afectada, y hace difícil una comparativa de capacidad. A pesar de esto, si se puede notar que la sección de la columna exterior del edificio de 4 pisos, tanto para material Hysteretic como Steel 02, fluye a la misma demanda de momento (80% o 90% de MP). Mientras que la columna interior fluye prácticamente en el valor de Mp.

En la estructura de 8 también se presenta un colapso. Sin embargo, su capacidad en los diagramas es más visible. Evidenciando así nuevamente que el material Hysteretic presenta más capacidad de deformación. De igual manera la fluencia en ambos materiales se produce prácticamente en los mismos valores. Siendo casi la misma para columna interior y exterior (70% de Mp).

La estructura de 12 pisos presenta la peculiaridad que en el material Hysteretic se reporta colapso, mientras que en el material Steel02 no. Sin embargo, en los diagramas del material Hysteretic sigue siendo visible la curva del comportamiento de capacidad. Con lo cual se puede constatar la misma tendencia que en las anteriores estructuras (el material Hysteretic presenta mayor capacidad de deformación, y los niveles de fluencia se dan en la misma demanda). La demanda de fluencia para esta estructura, tanto para material Steel02 como para Hysteretic, es mayor en la columna interior (Mp), a comparación con la exterior (70% de MP).

El ejemplar de 20 pisos presenta en ambos casos (material Hysteretic y Steel02), tanto para columna interior y exterior, el mismo comportamiento de lineal elástico ante el registro sísmico aplicado. Con la particularidad de que las columnas interiores reportan una exagerada estimación de sobre resistencia, siendo esta aproximadamente de 40% a 80% más que el Mp. Lo cual podría evidenciar un déficit en este tipo de modelación para edificios de gran altura, o que el material propuesto no sería el adecuado.

En la comparativa entre la distribución Gauss-Lobatto y Gauss-Radau Modificado. Se evidencia más que nada la distribución de los puntos de integración. Siendo la distribución GL más constante que la GRM.

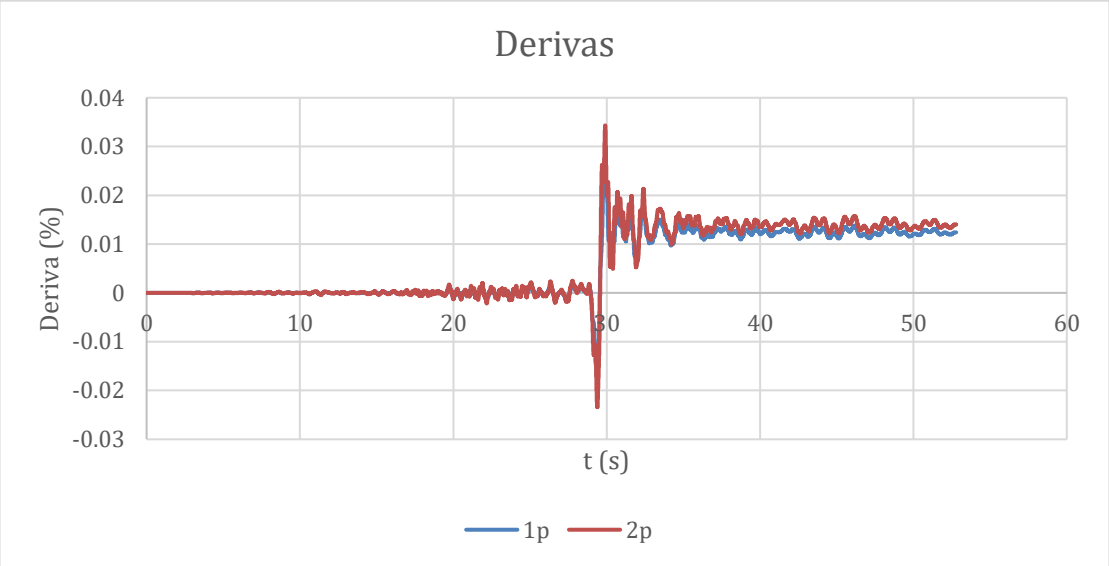
También se logra evidenciar que mientras más cerca se está a la conexión con la placa base, la demanda de capacidad tiende a incrementar. Siendo entonces la zona cerca a la conexión la que rige la ductilidad de la sección.

5.2.5. Derivas para Modelos con Longitud Plástica de L/12 con Material Steel02

A continuación, se muestra la variación de la deriva a través del tiempo, de cada piso. Estos resultados no son muy comunes de ver, por lo tanto se ha aprovechado al máximo las capacidades del Opensees.

Figura 5.64

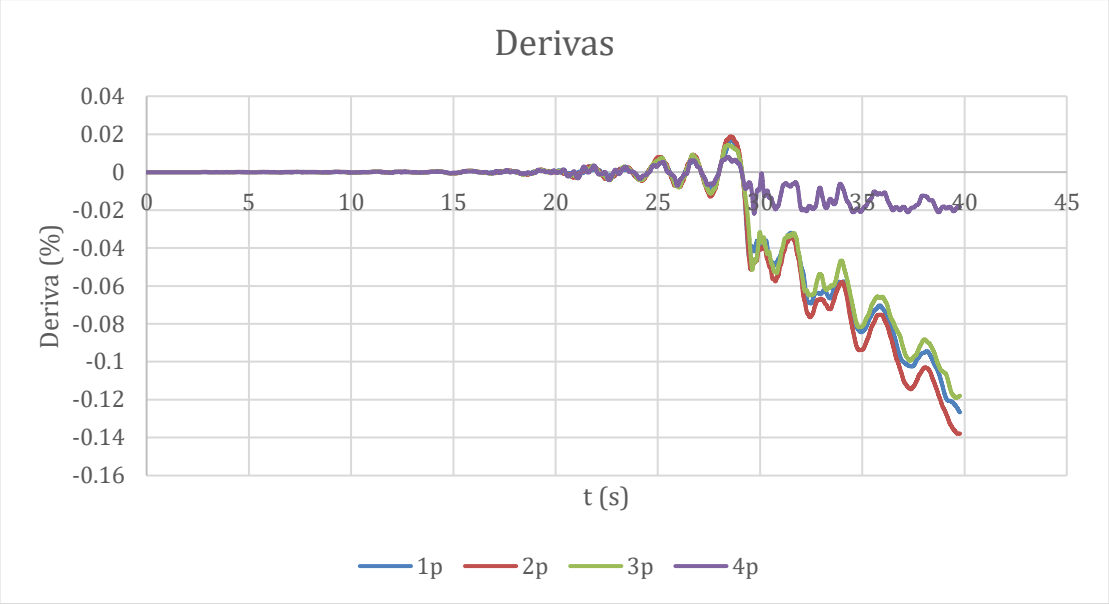
Variación de derivas a través del tiempo en la estructura de 2 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.65

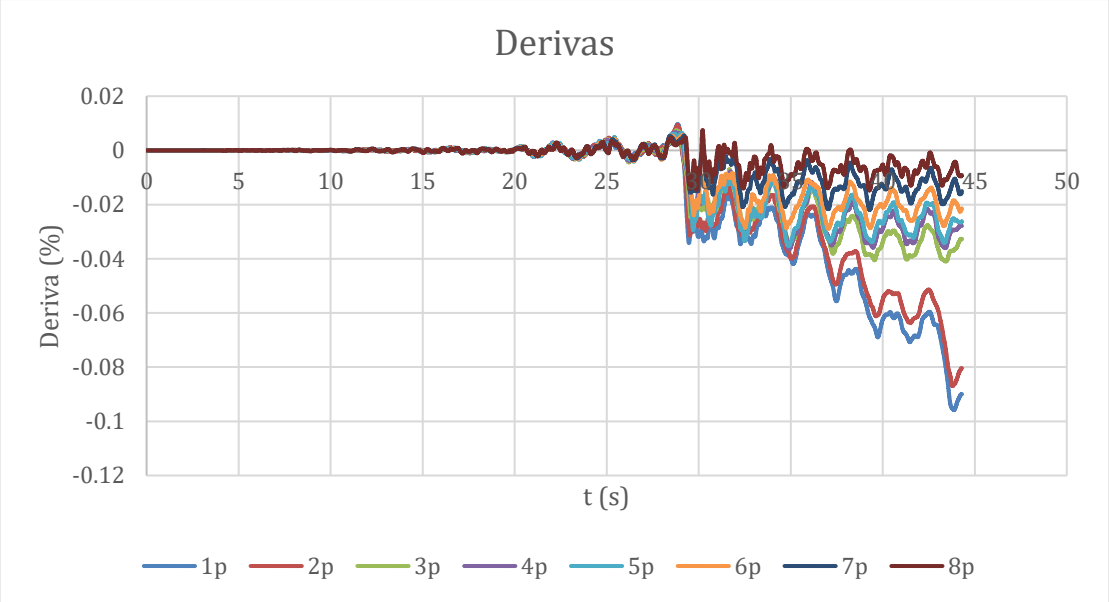
Variación de derivas a través del tiempo en la estructura de 4 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.66

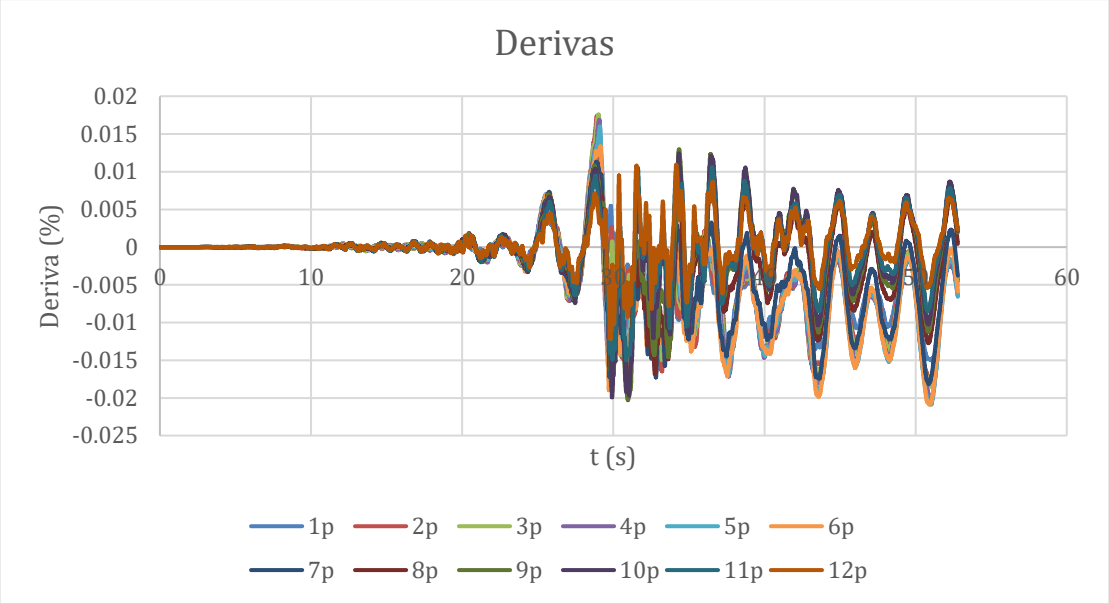
Variación de derivas a través del tiempo en la estructura de 8 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.67

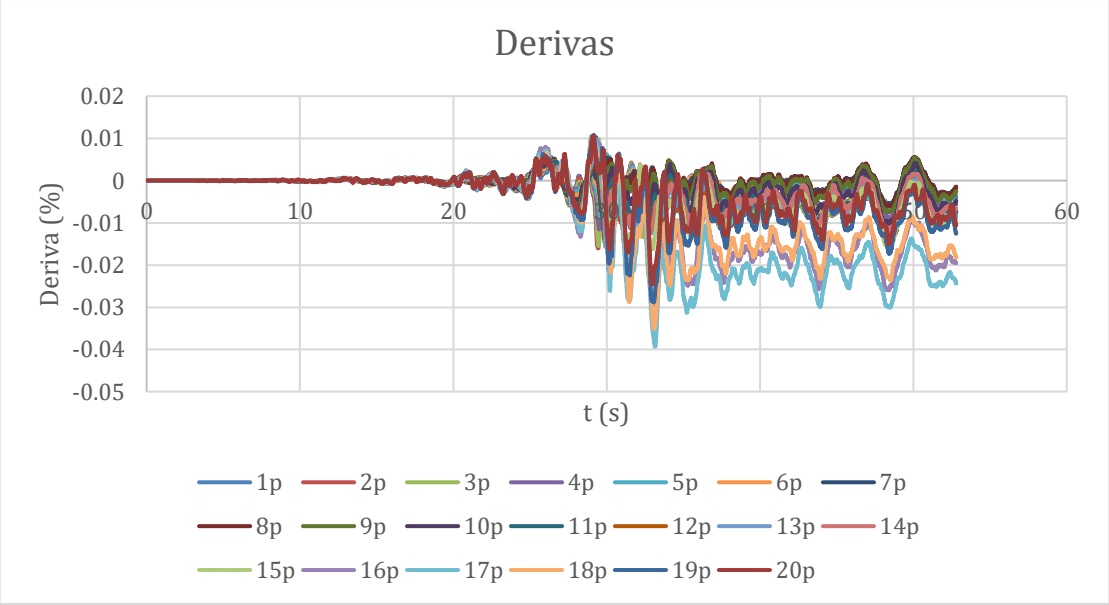
Variación de derivas a través del tiempo en la estructura de 12 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.68

Variación de derivas a través del tiempo en la estructura de 20 pisos



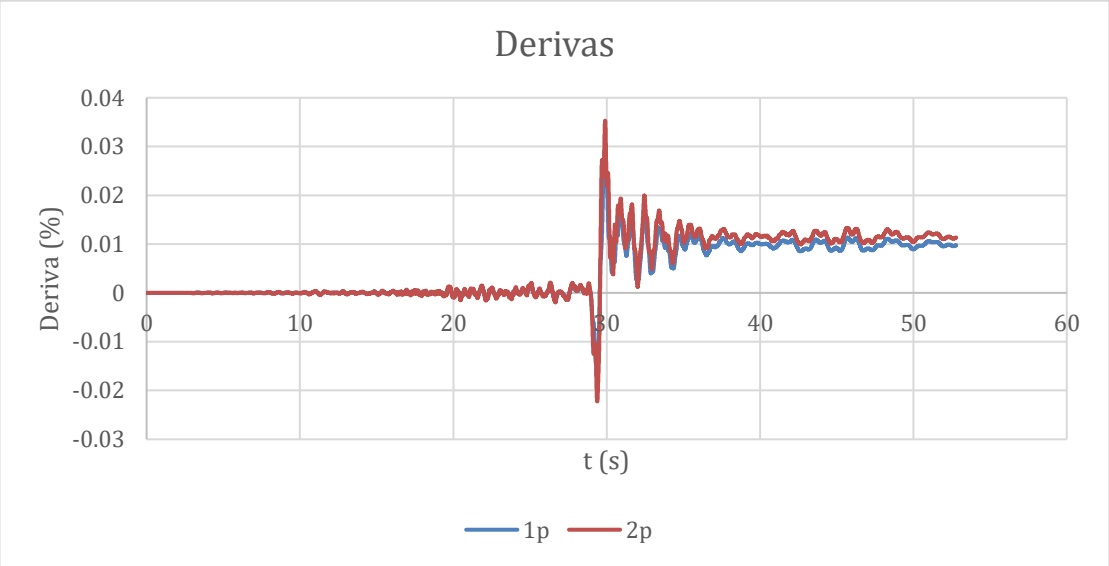
Nota. Las derivas corresponden a la deriva elástica

5.2.6. Derivas para Modelos con L. P. de L/12 con Material Hysteretic

De igual manera se realizó el análisis para el material Hysteretic y registrar comportamientos o tendencias destacables. A si mismo los gráficos a continuación muestran las derivas de todos los pisos en las estructuras correspondientes.

Figura 5.69

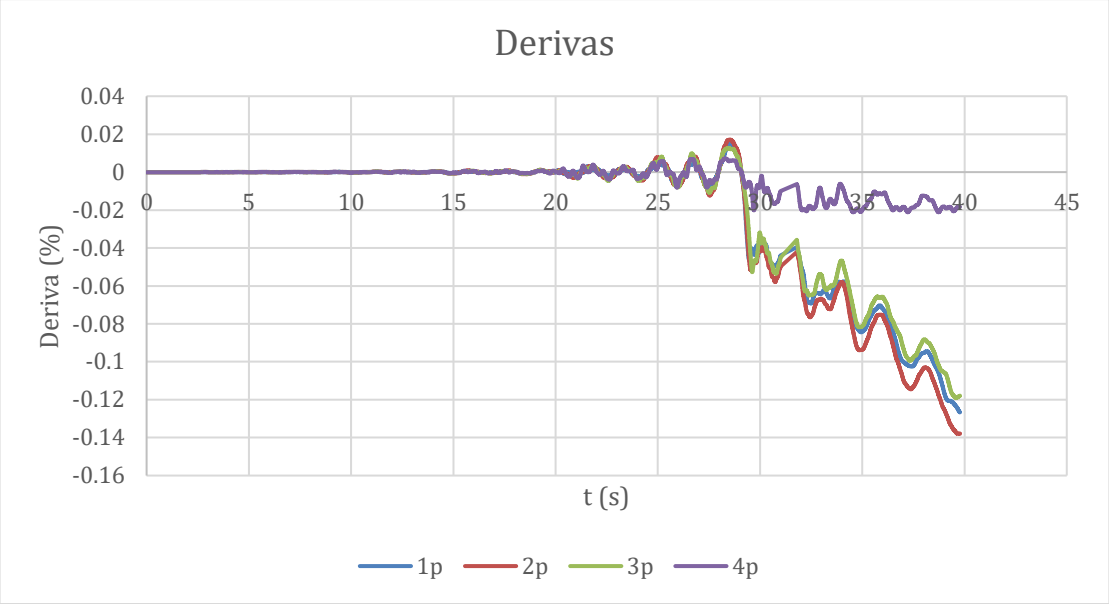
Variación de derivas a través del tiempo en la estructura de 2 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.70

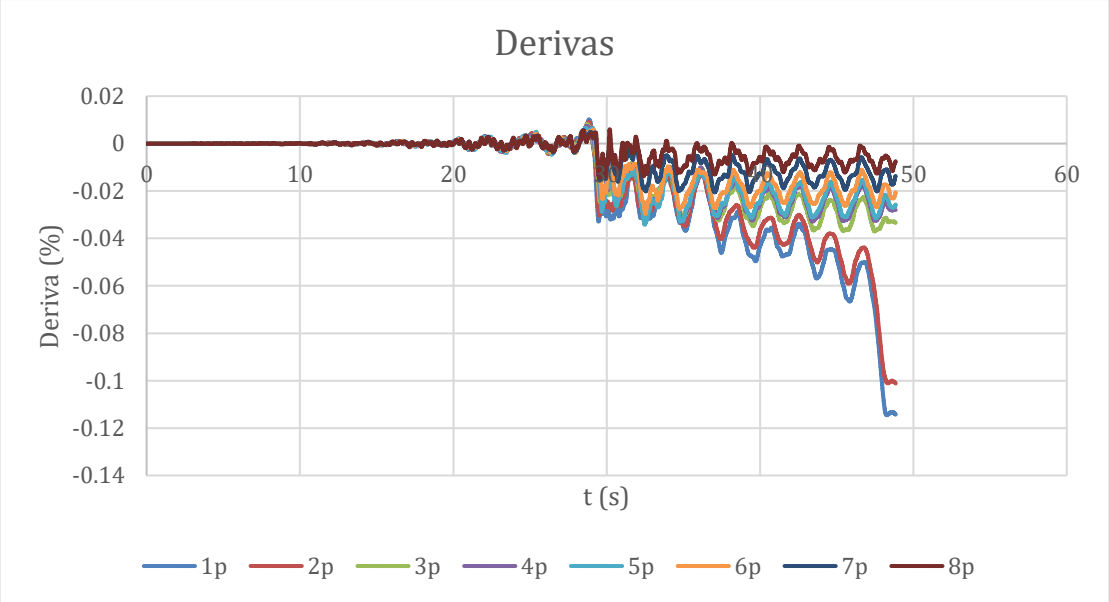
Variación de derivas a través del tiempo en la estructura de 4 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.71

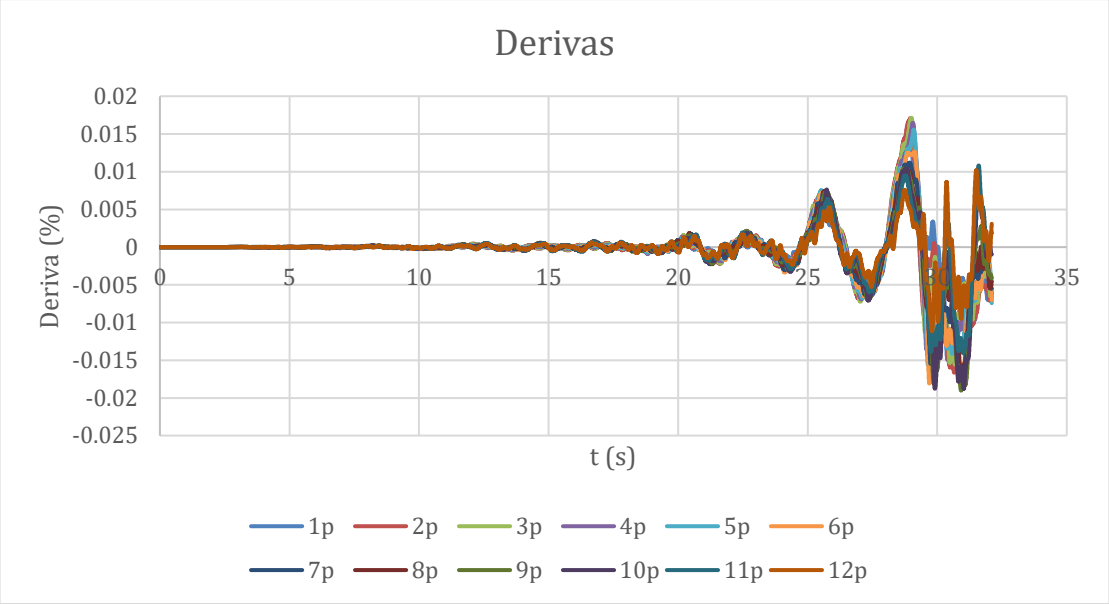
Variación de derivas a través del tiempo en la estructura de 8 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.72

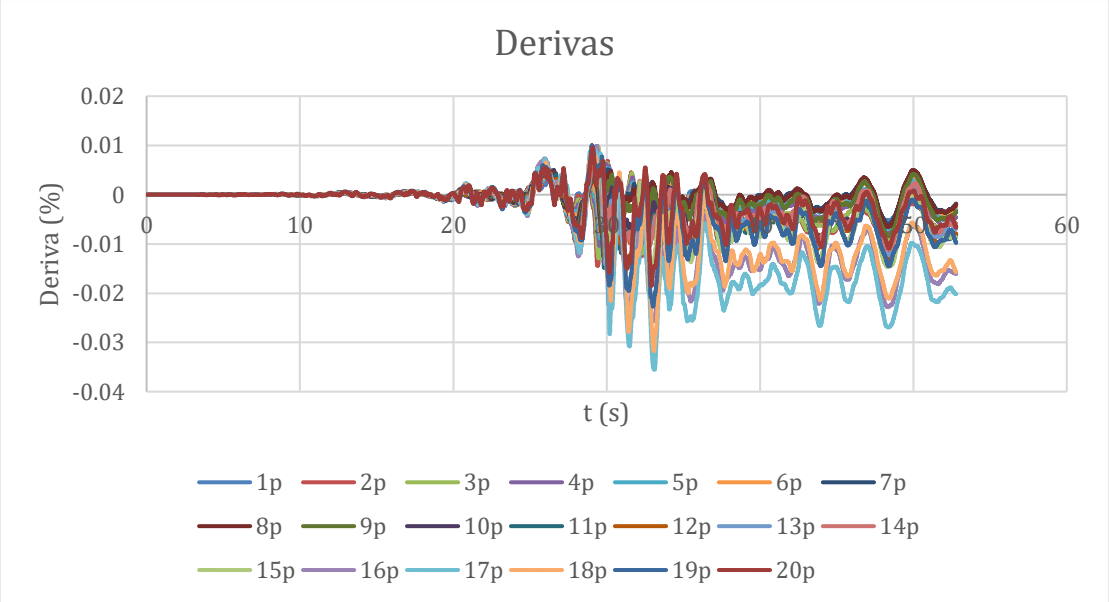
Variación de derivas a través del tiempo en la estructura de 12 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.73

Variación de derivas a través del tiempo en la estructura de 20 pisos



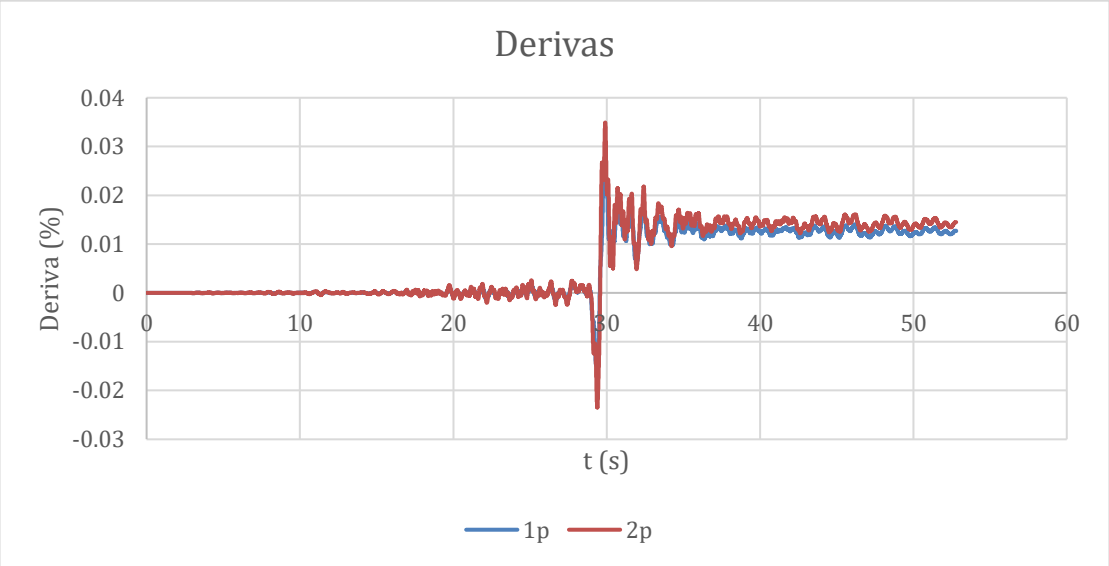
Nota. Las derivas corresponden a la deriva elástica

5.2.7. Derivas para Modelos con Longitud Plástica de Section Deepth (GL)

Los gráficos a continuación muestran la variación de la deriva a través del tiempo, por piso, y teniendo en cuenta el cambio de la longitud plástica a la altura de la sección. También se mantiene el material Steel02

Figura 5.74

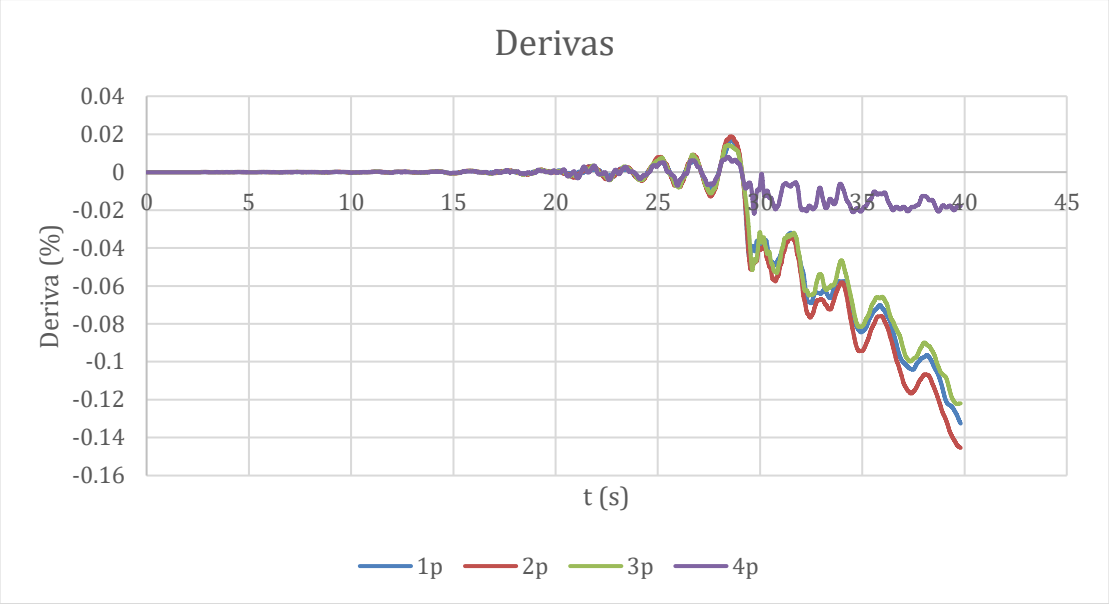
Variación de derivas a través del tiempo en la estructura de 2 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.75

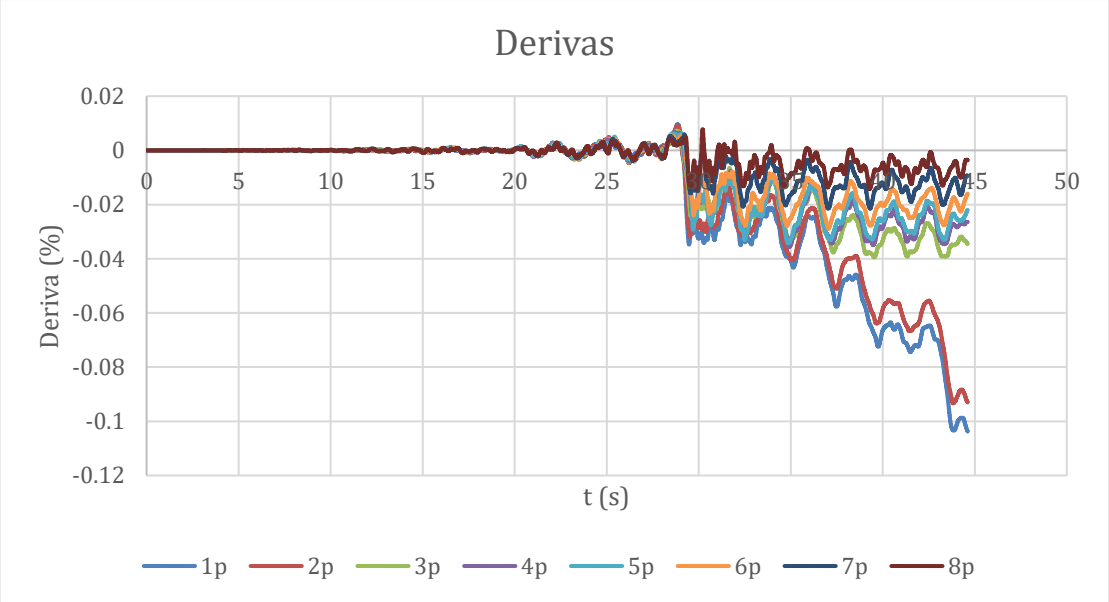
Variación de derivas a través del tiempo en la estructura de 4 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.76

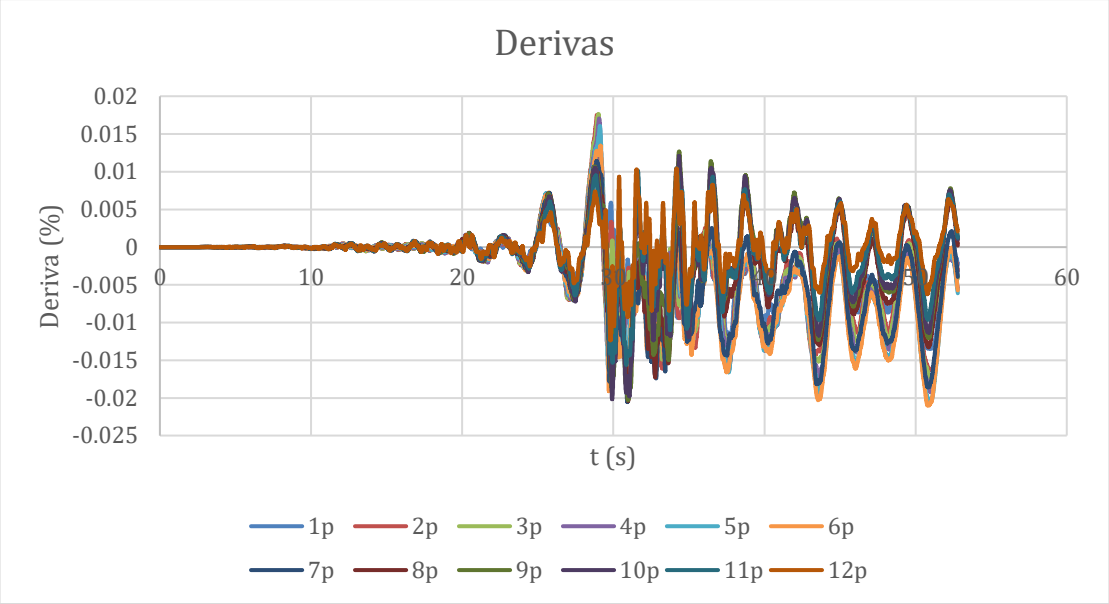
Variación de derivas a través del tiempo en la estructura de 8 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.77

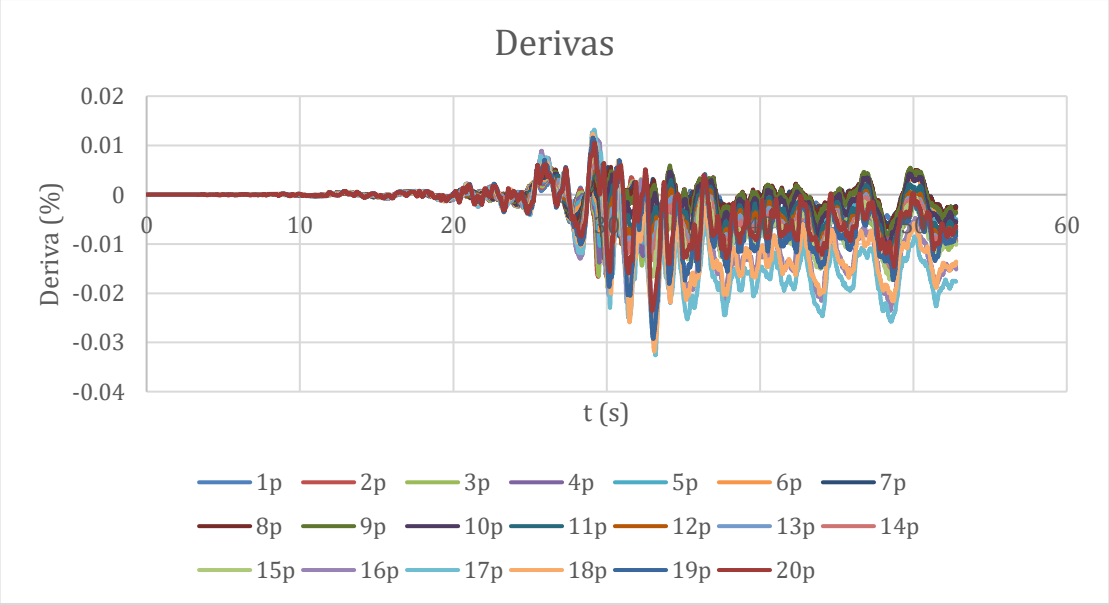
Variación de derivas a través del tiempo en la estructura de 12 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.78

Variación de derivas a través del tiempo en la estructura de 20 pisos



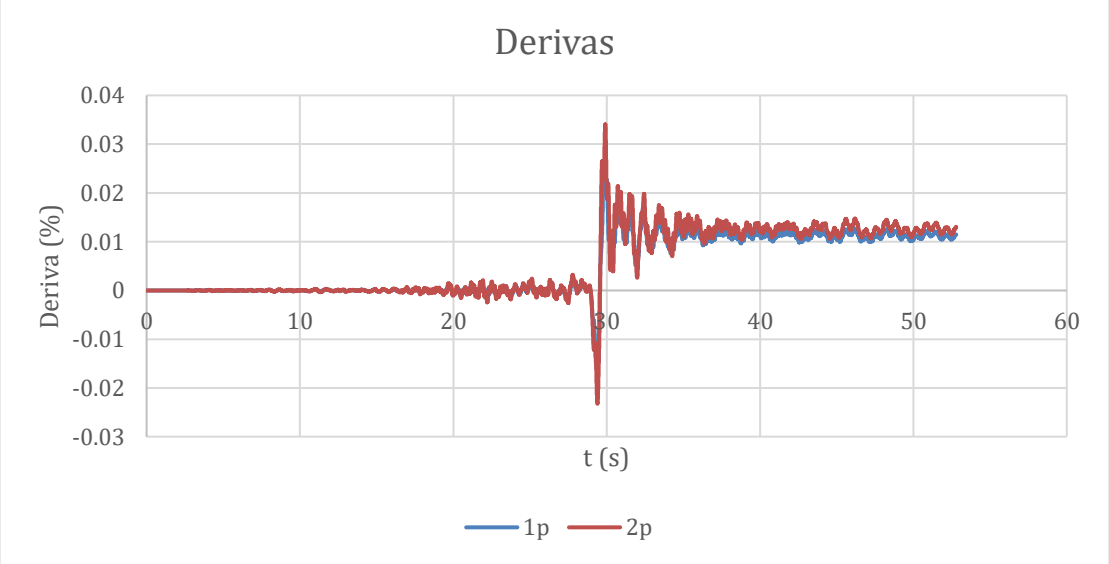
Nota. Las derivas corresponden a la deriva elástica

5.2.8. Derivas para Modelos con Longitud Plástica de 3/5(d) (GRM)

Los siguientes gráficos muestran como varía la deriva a través del tiempo, por cada piso de la estructura correspondiente. Teniendo en cuenta un material Steel02, y una longitud plástica muy similar a L/12 (3/5*setion deepth). Sin embargo, la disposición de puntos de integración es diferente, puesto que se usa distribución GRM

Figura 5.79

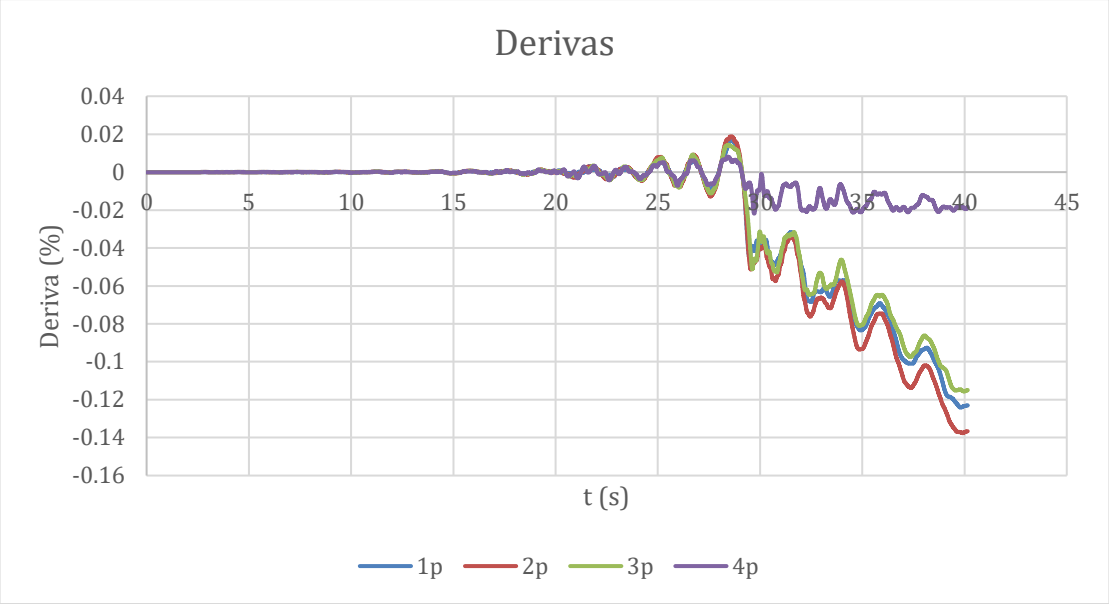
Variación de derivas a través del tiempo en la estructura de 2 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.80

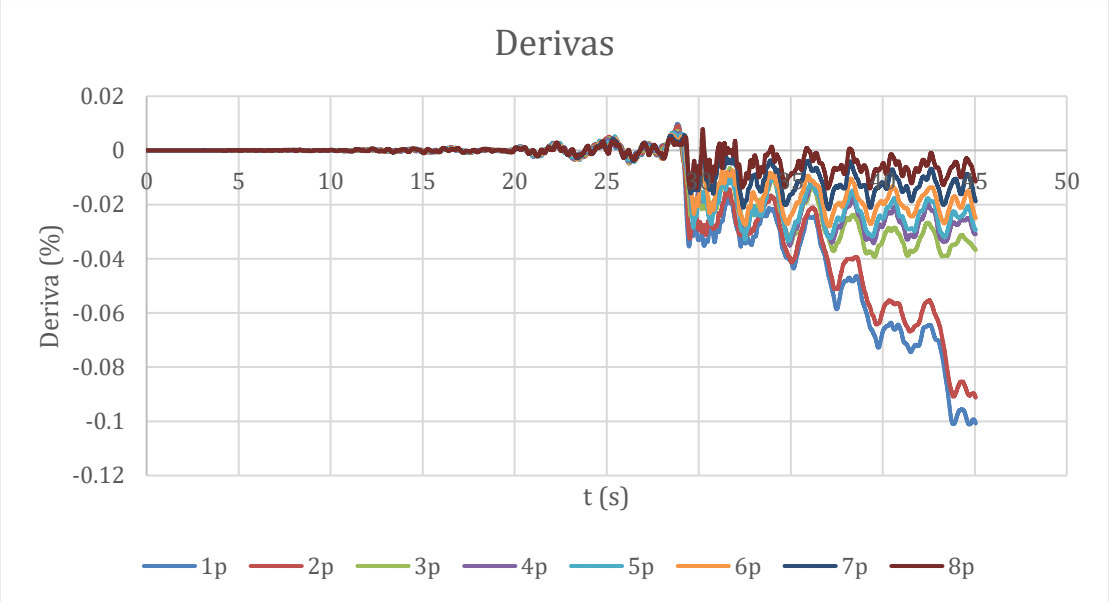
Variación de derivas a través del tiempo en la estructura de 4 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.81

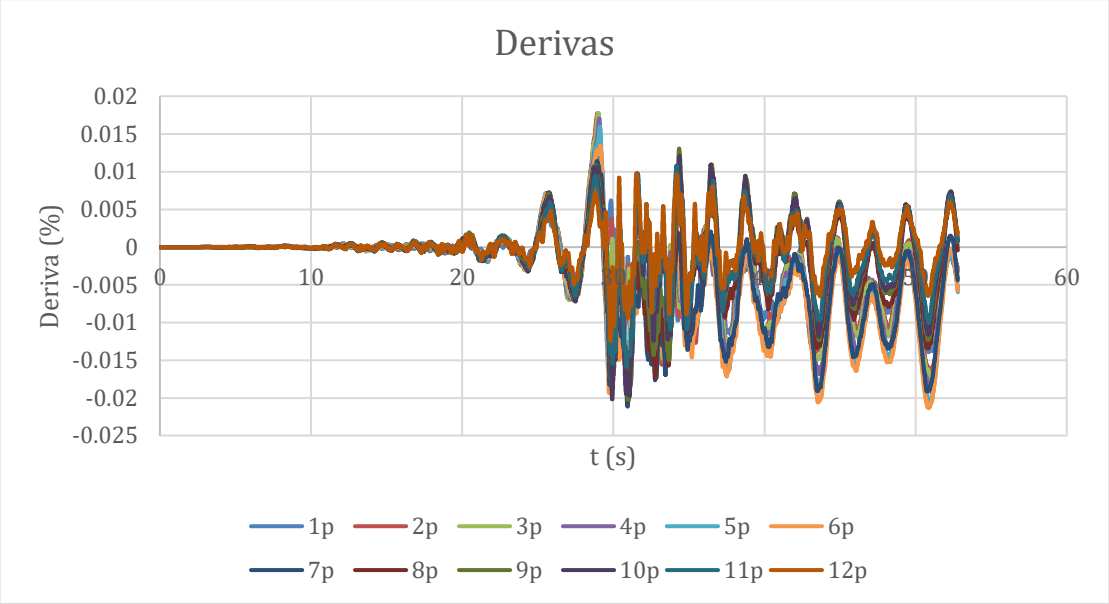
Variación de derivas a través del tiempo en la estructura de 8 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.82

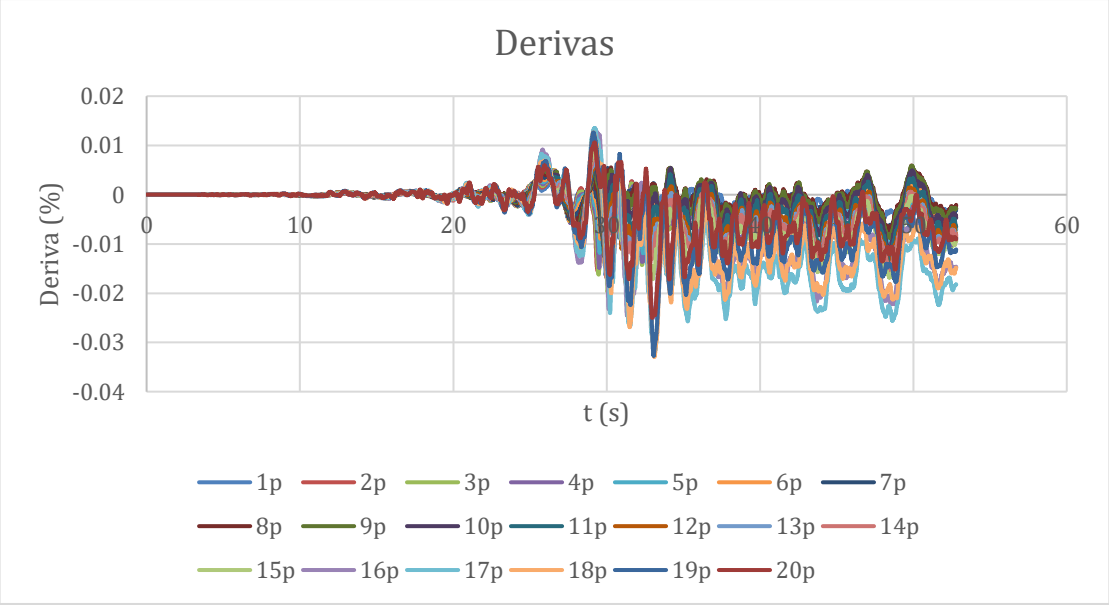
Variación de derivas a través del tiempo en la estructura de 12 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.83

Variación de derivas a través del tiempo en la estructura de 20 pisos



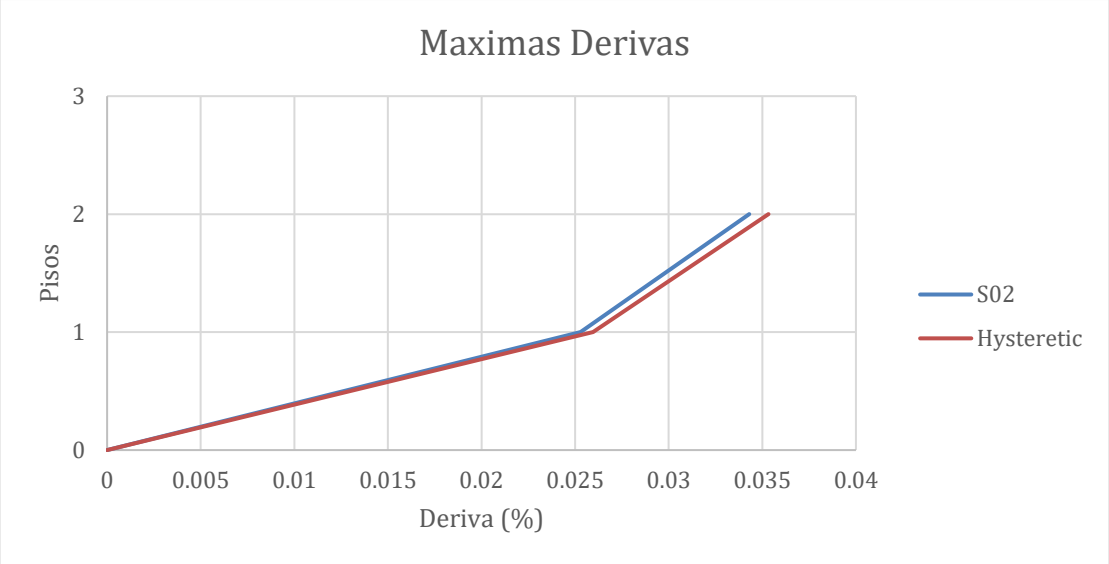
Nota. Las derivas corresponden a la deriva elástica

5.2.9. Comparación de Máximas Derivas por Material

A continuación, se hace la comparación del comportamiento de las máximas derivas de piso teniendo en cuenta el factor del material.

Figura 5.84

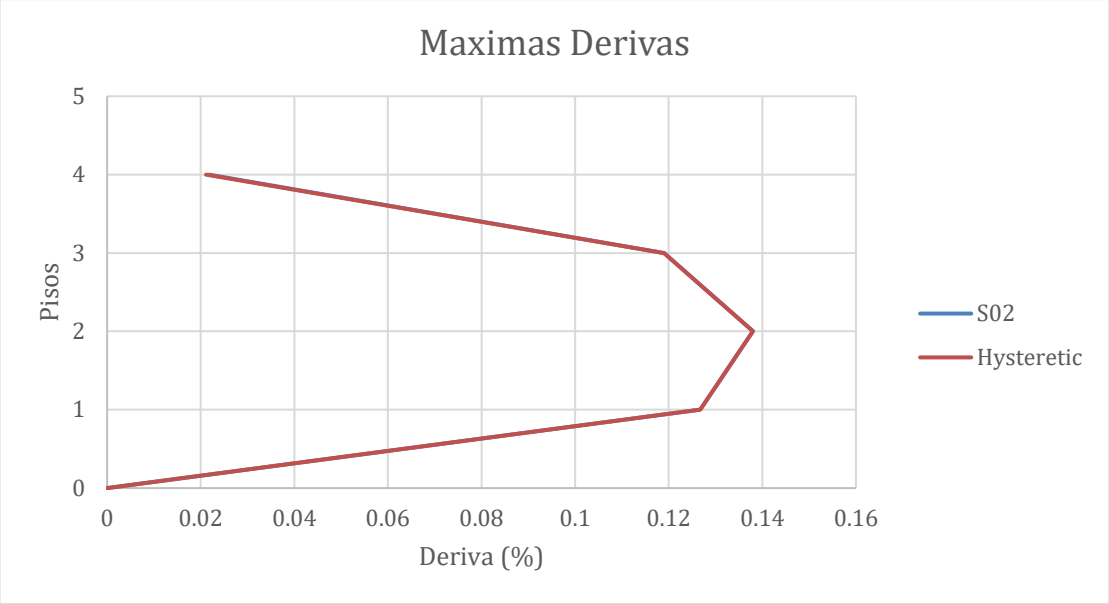
Estructura de 2 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.85

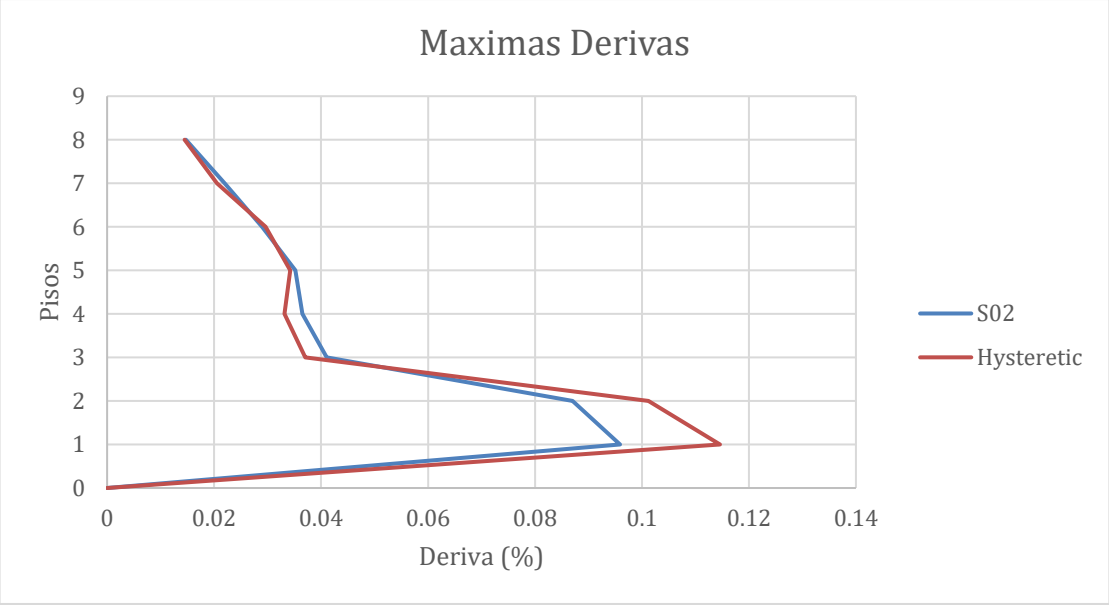
Estructura de 4 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.86

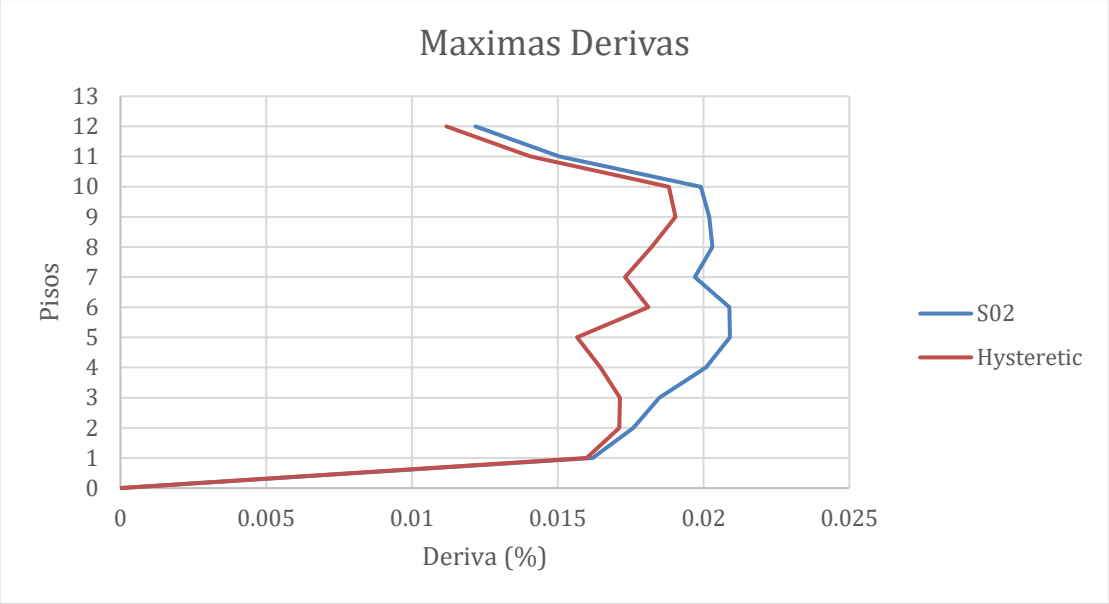
Estructura de 8 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.87

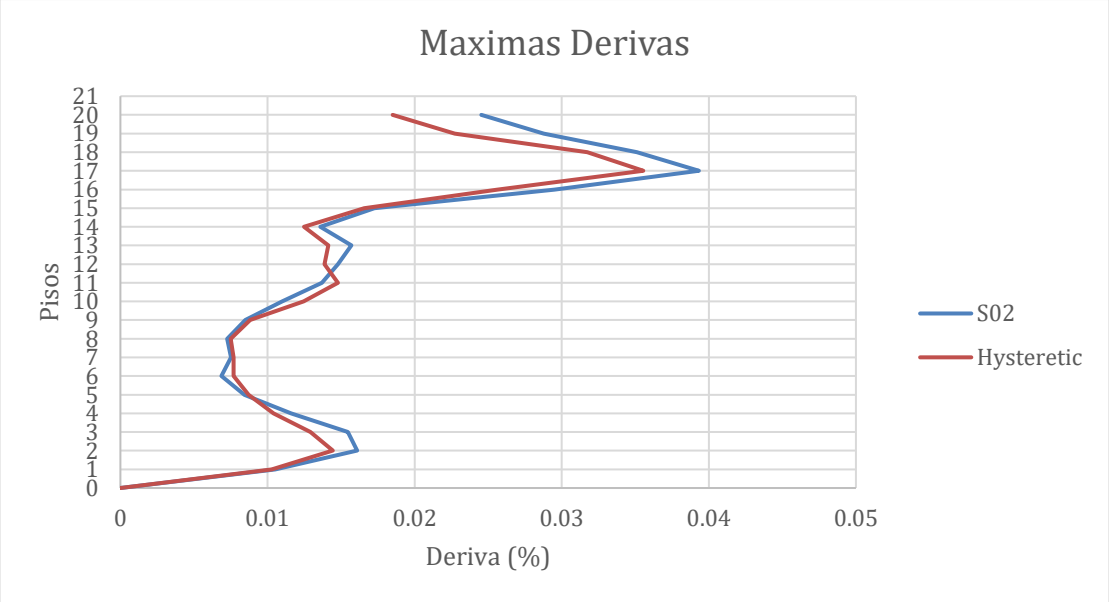
Estructura de 12 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.88

Estructura de 20 pisos



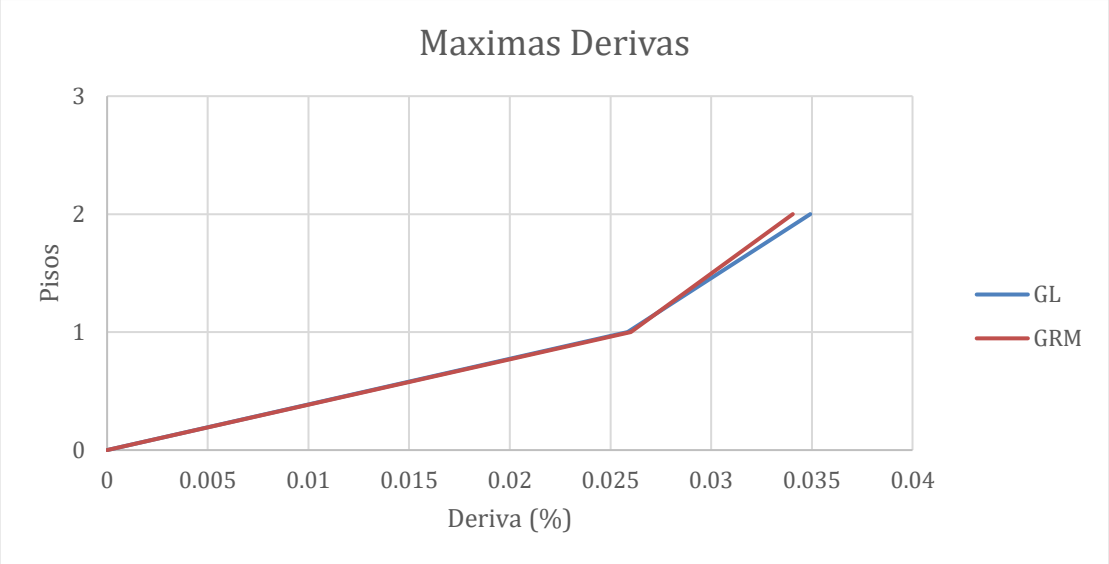
Nota. Las derivas corresponden a la deriva elástica

5.2.10. Comparación de Máximas Derivas por Tipos de Distribución

De igual manera se hace la comparación de las máximas derivas de piso, pero esta vez teniendo en cuenta el factor de distribución.

Figura 5.89

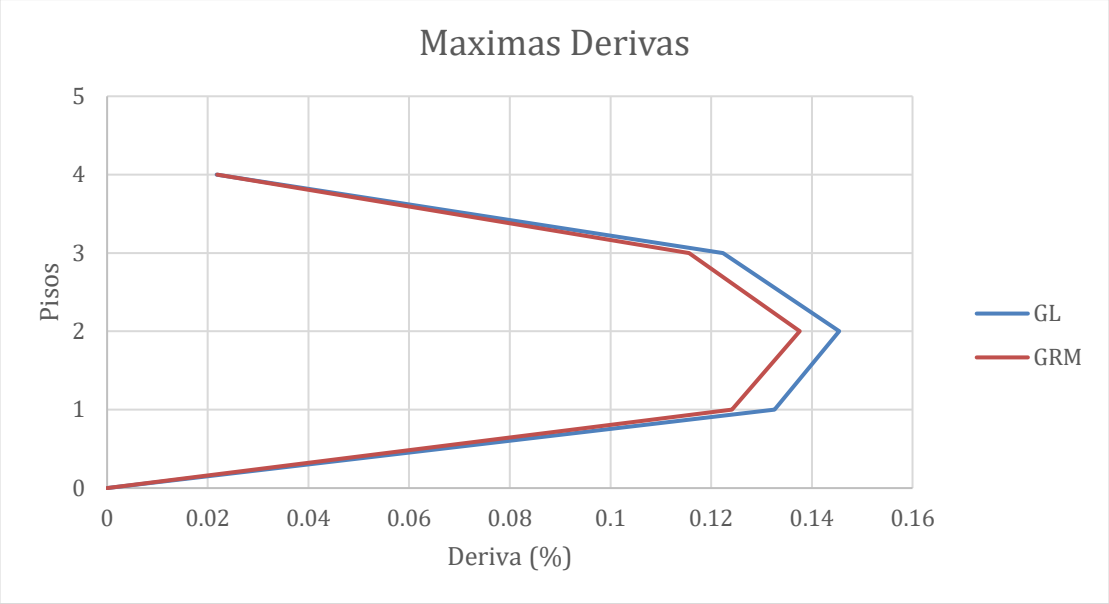
Estructura de 2 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.90

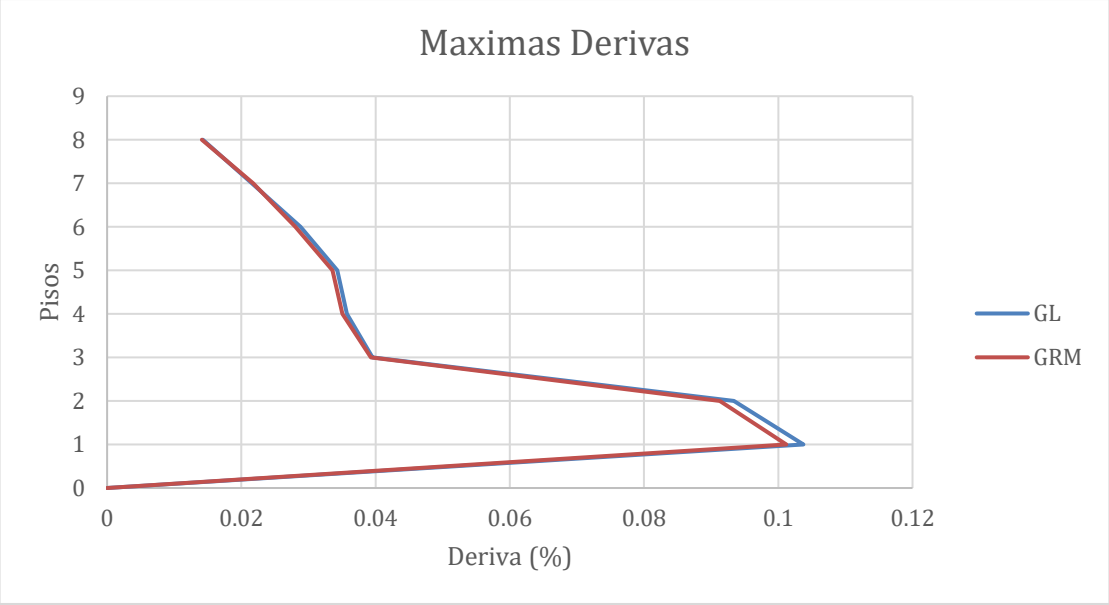
Estructura de 4 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.91

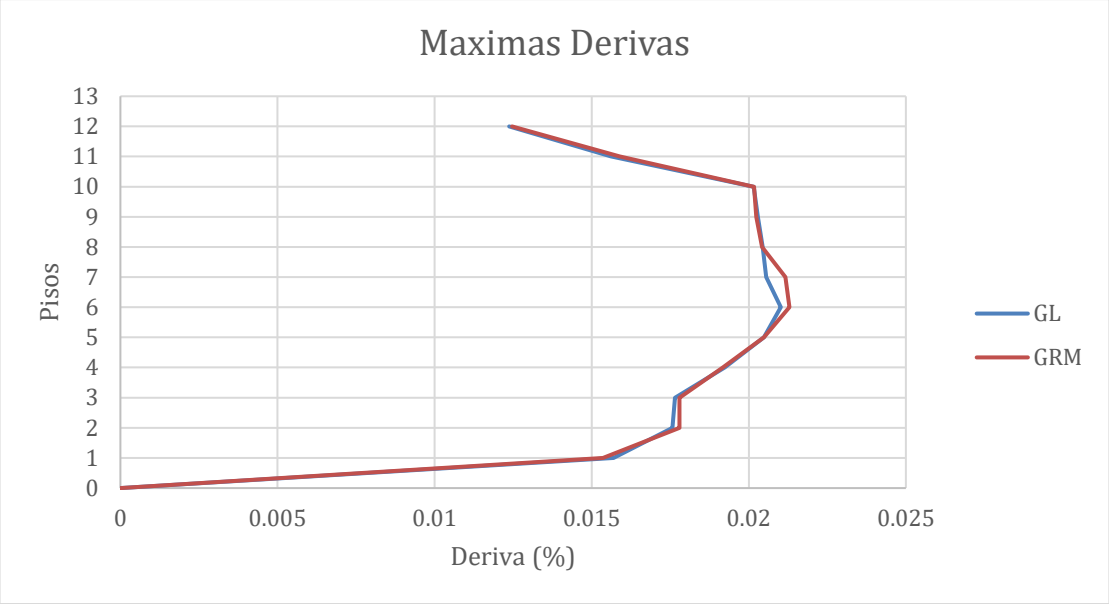
Estructura de 8 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.92

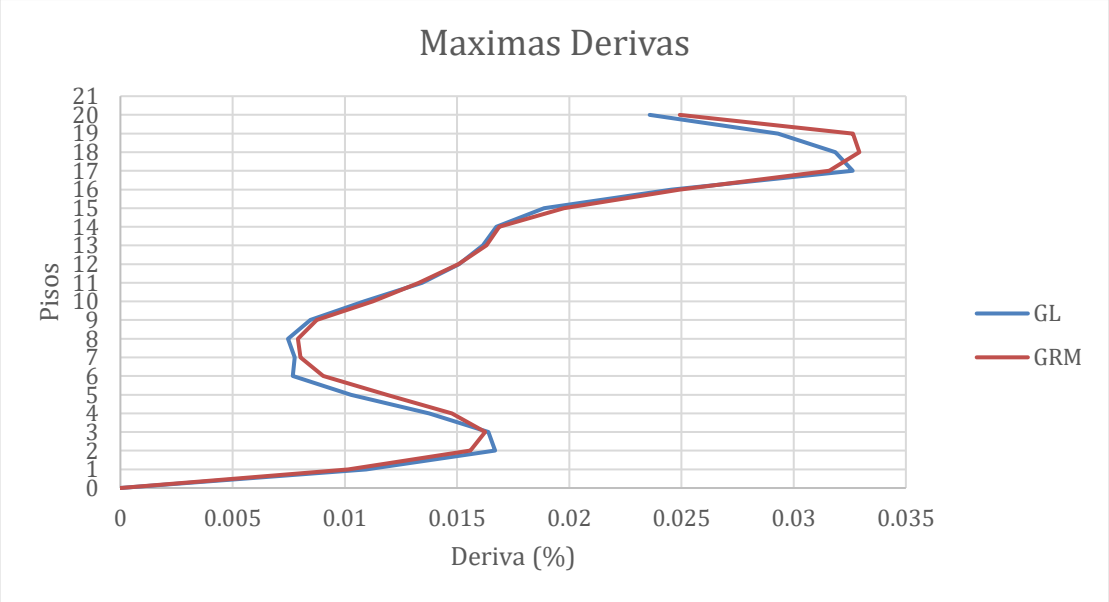
Estructura de 12 pisos



Nota. Las derivas corresponden a la deriva elástica

Figura 5.93

Estructura de 20 pisos



Nota. Las derivas corresponden a la deriva elástica

Al comparar las derivas tanto por material como por tipos de distribución, se logra apreciar la misma tendencia en todas las estructuras. La cual es que la deriva aumenta significativamente a partir de los 30 segundos del evento. Esto tiene mucho sentido puesto que, el registro muestra su pico en ese tiempo. Sin embargo, en estructuras de 4 y 8 pisos se observa una falla de entrepiso débil, puesto que las derivas de primer y segundo piso (en caso de la estructura de 4 pisos se incluye hasta el tercero) son sustancialmente mayores en comparación con los demás pisos.

En edificios altos, de 12 y 20 pisos se puede evidenciar una mayor demanda en pisos superiores, y no en el primer piso. En el edificio de 2 pisos se registra una demanda máxima de 3.5%, en el de 4 pisos de casi del 14%, y en el de 8 pisos cerca del 11%. Para el de 12 pisos se tiene una demanda de 2.1%, y para el de 20 pisos una demanda cercana al 3.9%. Por consiguiente, es notable el colapso de las estructuras de 4 y 8 pisos.

No se nota ningún patrón de comportamiento específico por cambio de materiales. Sin embargo, los resultados en ambos se pueden estimar que son parecidos. Salvo por algunos casos en los que se puede observar una mayor demanda en material Hysteretic, y en otros por el material Steel02. En cuanto a cambio de distribución no se encuentran diferencias y los valores son muy parecidos en todos los casos.

CAPITULO 6.- CONCLUSIONES

Con base en los análisis estáticos no lineales (NSP “siglas en inglés”), se pudo desestimar ciertos modelos de estudio y propuestas de materiales hechas por diversos autores. Esto debido a que se tuvo información de estudios previos, pero con modelos de plasticidad concentrada (Falborski, Torres-Rodas, Zareian, & Kanvinde, 2020), y bibliografías de modelos con fibras cuyos resultados eran muy parecidos a los CPH. Por lo tanto, el parámetro más importante evaluado era el que las curvas de capacidad presenten el mismo comportamiento que las CPH. Por consiguiente, a pesar de que los resultados de los modelos “Fiber Hinge” y “Fiber” correspondientes a los propuestos por Pozo et al., 2018 presentan el mismo comportamiento que en su estudio, estos no se acoplan a las curvas CPH, puesto que se observa una elevada capacidad dúctil en las estructuras. Esto podría llevar a una sobre estimación en la capacidad de la estructura. Con esto en mente, entonces los estudios se redujeron a los modelos “Fiber Hinge and CPH” y “Fiber and CPH”, dado que la variación de resultados no era significativa, se decidió usar solo uno de estos para los análisis dinámicos.

En cuanto a material, se desestimó el uso del material en paralelo propuesto por Pozo et al., 2021 en revisión, puesto que ningún modelo logró converger. Con respecto al material Hysteretic, si bien es cierto que los modelos convergieron, la demanda computacional es mucho mayor a la del material Steel02. Por lo tanto, por cuestiones prácticas también se lo desestimó.

Debido a que en las bibliografías se establecían integraciones tipo Gauss Radau Modificado, entonces se probó también con tipo de integración Gauss-Lobatto, puesto que la integración GL es más sencilla su definición. Aparte que se tiene un mayor control en el número de puntos de integración.

Por lo tanto, la recomendación de modelación para NSP incluyendo efectos P-Δ sería la de modelos “Fiber Hinge and CPH”, con material Steel02, y distribución Gauss-Lobatto como tipo de integración de fibras.

Si bien es cierto que el material Steel02 muestra los mejores resultados, los modelos de 20 pisos no lograron adaptarse por completo a la curva objetivo (CPH). Sumado a esto la exagerada sobre resistencia evidenciada en los análisis dinámicos por parte de las columnas interiores (en algunos casos cerca del 80% del Mp). Se puede establecer entonces, que el tipo de modelación tipo “Fiber Hinge and CPH” con material Steel02 obtiene resultados confiables hasta edificios de 12 pisos. Y que, para edificios de mayor número de pisos, la confiabilidad de sus resultados decrece. Por consiguiente, edificios altos (mayores a 20 pisos) necesitan un estudio focalizado de material y modelación, para la implementación de fibras.

También queda en evidencia el deterioro de la capacidad en las estructuras de acero al incluir efectos P-Δ en el NSP. Sin embargo, normativamente no es obligatorio incorporar efectos P-Δ en el NSP. Lo cual puede conllevar a sobreestimaciones en la capacidad real de la estructura, provocando así un mayor daño del que se tenía previsto.

CAPITULO 7.- REFERENCIAS

ASTM. (2004). A 992: Standard Specification for Structural Shapes. Conshohocken, USA.

Astudillo Carpio, B. X. (Octubre de 2018). Modelación y análisis por desempeño de una estructura de acero, considerando deterioro para la predicción del colapso. Cuenca, Azuay, Ecuador.

Bosco, M., Ferrara, E., Ghersi, A., Marino, E., & Rossi, P. P. (2014). Improvement of the Model Porposed by Menegotto and Pinto for Steel. European Conference on Earhquake Engineering and Seimology(2), 25-29.

Falborski, T., Torres-Rodas, P., Zareian, F., & Kanvinde, A. (2020). Effect of base-connection strength and ductility on the seismic performance of steel moment-resisting frames. *Journal of Structural Engineering*.

FEMA. (2009). FEMA P440A. *Effects of Strength and Stiffness Degradation on Seismic Response.* California, USA.

Filippou, F. C., Spacone, E., & Taucer, F. F. (1996). FIBRE BEAM-COLUMN MODEL FOR NON-LINEAR ANALYSIS OF R/C FRAMES: PART I. FORMULATION. California, USA: John Wiley & Sons, Ltd.

Lee, J., Engelhardt, M. D., & Jeong Choi, B. (2015). Constitutive Model for ASTM A992 Steel at Elevated Temperature. Internationa Journal of Steel Structures, 3(15), 733-741.

Martinez Pesantez, I. L., & Pozo Ocampo, P. S. (Octubre de 2018). Análisis por desempeño de una estructura especial de acero modelada con fibras controladas por fuerzas en el software OpenSees. Cuenca, Azuay, Ecuador.

Mora, D., & Aguiar, R. (2015). MODELACIÓN DE DIAGRAMA MOMENTO-CURVATURA Y MOMENTO-ROTACIÓN EN SECCIONES DE ACERO ESTRUCTURAL. *CIENCIA*, 26.

Nguyen, A. R. (2017). Comparative Spectral Analysis of Flexible Structure Models: the Euler-Bernoulli Beam model, the Rayleigh Beam model, and the Timoshenko Beam Model. Durham, New Hampshire, USA.

NIST. (2010). Evaluation of the FEMA P-695 Methodology for Quantification of Building Seismic Performance Factors. Redwood City, California, USA.

NIST. (Abril de 2017). Guidelines for Nonlinear Structural Analysis for Design of Buildings Part IIa – Steel Moment Frames. Redwood City, California, USA.

Orakcal, K., & Wallace, J. W. (2006). *Flexural Modeling of Reinforced Concrete Walls—Experimental Verification.* ACI Structural Journal, 103(2), 196-206.

Powell, G. H. (2010). Modeling for Structural Analysis. California, USA.

Pozo, S., Astudillo, B., Samaniego, E., & Flores, F. (2020). REGULARIZATION METHOD TO INCLUDE MATERIAL SOFTENING IN FIBER BEAM COLUMN ELEMENTS FOR SEISMIC PERFORMANCE ASSESMENT OF STEEL FRAMES. *EURODYN 2020* (pág. 16). Atenas: EASD.

Pozo, S., Astudillo, B., Samaniego, E., & Flores, F. (2021 en revisión). Objective Phenomenological Constitutive Law for Collapse Analyses in Distributed Plasticity Steel Frame Models. Cuenca, Azuay, Ecuador.

Scott, M. H. (26 de Enero de 2011). Numerical Integration Options for the Force-Based Beam-Column Element in OpenSees. Oregon, USA.

The Mathaworks, Inc. (1994-2021). *MathWorks*. Obtenido de Mathworks: https://www.mathworks.com/videos/matlab-overview-61923.html

University of Berkeley. (2006). *Opensees*. Obtenido de Opensees: http://www.opensees.berkeley.edu

ÍNDICE DE ANEXOS

Enlace de Descarga de Anexos

https://mega.nz/folder/LlIhhaBI#Kk2hrv1UkkvpPt5EWq8kcw

ANEXO A1: CODIGO MATLAB DE EJECUCIÓN PARA MODELOS L/12 FIBER HINGE

Para cambiar el número de pisos solo basta con cambiar el NumS. También se requiere de un Archivo txt (AISC.txt), en el cual se encuentren redactadas todas las tablas del AISC con las propiedades de las secciones

```
clear all;
close all;
clc;

cd ('E:\maestría usfq estructuras\Tesis USFQ\modelos Lentre12\cws 5 10 5\G-L NP 3\s02\Fiber Hinge\runner 8')

%Structural configuration:

NumS=8;     %Number of stories of the building
bay=240;    %Bay length
h1=180;     %Height of the first story
hi=156;     %Height of the rest of stories
Ry=1.1;
NumB=3;     %Number of bays without leaning column
%PROPERTIES OF THE PLASTIC HINGES:
%type=2 GRAVITY ANALYSIS!!
%type=1 REDUCTION CAPACITY OF THE COLUMN !!

type=2;

%Member sizes:

%Input the size of the member for each story from bottom to top

switch NumS
    case 2

    %TWO STORY BUILDING:

    beam_sizes = {'W30X132', 'W16X31'}';
    extcol_sizes = {'W24X131', 'W24X131'}';
    incol_sizes = {'W24X162', 'W24X162'}';

    ext_dblplate = [0.0, 0.0]';
    int_dblplate = [0.0, 0.0]';

    case 4

    %FOUR STORY BUILDING:

    beam_sizes = {'W21X73', 'W21X73', 'W21X57', 'W21X57'}';
    extcol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';
```

```
    incol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0]';
    int_dblplate = [5/16, 5/16, 5/16, 5/16]';

    case 8
    %EIGHT STORY BUILDING:

    beam_sizes = {'W30X108', 'W30X116', 'W30X116', 'W27X94',
'W27X94','W27X94','W24X84','W21X68'}';
    extcol_sizes = {'W24X131', 'W24X131', 'W24X131',
'W24X131','W24X131','W24X131','W24X131','W24X94'}';
    incol_sizes = {'W24X162', 'W24X162', 'W24X162',
'W24X162','W24X162','W24X131','W24X131','W24X94'}';


    ext_dblplate = [1/16, 1/16, 1/16, 0.0, 0.0,  0.0,  0.0,  0.0]';
    int_dblplate = [9/16, 3/8, 11/16, 3/8, 9/16, 7/16, 9/16, 5/16]';

    case 12
    %TWELVE STORY BUILDING:

    beam_sizes = {'W30X124', 'W30X132', 'W30X132', 'W30X132',
'W30X116','W30X116','W30X116','W30X116','W27X94','W27X94','W24X84','W24X84'
}';
    extcol_sizes = {'W24X207', 'W24X207', 'W24X207', 'W24X162', 'W24X162',
'W24X146', 'W24X146', 'W24X131', 'W24X131', 'W24X131', 'W24X131',
'W24X84'}';
    incol_sizes =  {'W24X207', 'W24X207', 'W24X207', 'W24X207', 'W24X207',
'W24X176', 'W24X176', 'W24X162', 'W24X162', 'W24X131', 'W24X131',
'W24X94'}';

    ext_dblplate = [0.0, 0.0, 1/16, 1/16, 0.0, 0.0, 1/16, 1/16, 0.0, 0.0,
1/16, 1/16]';
    int_dblplate = [1/2, 7/16, 5/8, 5/8, 5/8, 5/8, 11/16, 11/16, 9/16,
9/16, 9/16, 9/16]';

    case 20
    %TWENTY STORY BUILDING:

    beam_sizes = {'W33X169', 'W33X169', 'W33X169', 'W33X169', 'W33X169',
'W33X169', 'W33X169', 'W33X169', 'W33X141', 'W33X141', 'W33X141',
'W33X141', 'W33X141', 'W33X141', 'W30X108', 'W30X108', 'W30X108',
'W30X108', 'W24X62', 'W24X62'}';
    extcol_sizes = {'W14X426', 'W14X426', 'W14X426', 'W14X426', 'W14X426',
'W14X398', 'W14X398', 'W14X370', 'W14X370', 'W14X311', 'W14X311',
'W14X283', 'W14X283', 'W14X233', 'W14X233', 'W14X159', 'W14X159',
'W14X132', 'W14X132', 'W14X132'}';
    incol_sizes =  {'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X335',
'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X279', 'W24X279',
'W24X250', 'W24X250', 'W24X250', 'W24X250', 'W24X162', 'W24X162',
'W24X162', 'W24X162', 'W24X103'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1/16, 1/16, 1/4, 1/4, 3/8, 3/8, 0.0, 0.0]';
    int_dblplate = [1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 5/16, 5/16,
1/2, 1/2, 1/2, 1/2, 9/16, 9/16, 9/16, 9/16, 3/16, 3/16 ]';
```

```
end

%Material properties:

Fy = 55; % Nominal yield point (ksi)
E_steel = 29000; % ksi
nu_steel = 0.3;
G_steel = E_steel/(2*(1+nu_steel)); % ksi
%alpha = 0.03; % hardening in third segment of trilinear panel zone
material

%Read AISC.text file which contain mechanical properties of the cross
sections

AISC_ID = fopen('AISC.txt');
AISC_data = textscan(AISC_ID, '%s %f %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f');

beam_A      = zeros(NumS,1); % area
beam_d      = zeros(NumS,1); % depth
beam_bf     = zeros(NumS,1); % flange length
beam_tw     = zeros(NumS,1); % web thickness
beam_tf     = zeros(NumS,1); % flange thickness
beam_bf_2tf = zeros(NumS,1); % bf/2tf
beam_h_tw   = zeros(NumS,1); % h/tw
beam_Ix     = zeros(NumS,1);
beam_Zx     = zeros(NumS,1);
beam_Sx     = zeros(NumS,1);
beam_rx     = zeros(NumS,1);
beam_Iy     = zeros(NumS,1);
beam_Zy     = zeros(NumS,1);
beam_Sy     = zeros(NumS,1);
beam_ry     = zeros(NumS,1);
beam_J      = zeros(NumS,1);
beam_Aweb   = zeros(NumS,1); % equivalent to (d-2tf)*tw

extcol_A      = zeros(NumS,1);
extcol_d      = zeros(NumS,1);
extcol_bf     = zeros(NumS,1);
extcol_tw     = zeros(NumS,1);
extcol_tf     = zeros(NumS,1);
extcol_bf_2tf = zeros(NumS,1);
extcol_h_tw   = zeros(NumS,1);
extcol_Ix     = zeros(NumS,1);
extcol_Zx     = zeros(NumS,1);
extcol_Sx     = zeros(NumS,1);
extcol_rx     = zeros(NumS,1);
extcol_Iy     = zeros(NumS,1);
extcol_Zy     = zeros(NumS,1);
extcol_Sy     = zeros(NumS,1);
extcol_ry     = zeros(NumS,1);
extcol_J      = zeros(NumS,1);
extcol_Aweb   = zeros(NumS,1);

incol_A      = zeros(NumS,1);
incol_d      = zeros(NumS,1);
```

```
incol_bf     = zeros(NumS,1);
incol_tw     = zeros(NumS,1);
incol_tf     = zeros(NumS,1);
incol_bf_2tf = zeros(NumS,1);
incol_h_tw   = zeros(NumS,1);
incol_Ix     = zeros(NumS,1);
incol_Zx     = zeros(NumS,1);
incol_Sx     = zeros(NumS,1);
incol_rx     = zeros(NumS,1);
incol_Iy     = zeros(NumS,1);
incol_Zy     = zeros(NumS,1);
incol_Sy     = zeros(NumS,1);
incol_ry     = zeros(NumS,1);
incol_J      = zeros(NumS,1);
incol_Aweb   = zeros(NumS,1);

%Load cross sections properties of each story

for i = 1:NumS

beam_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);

end

for i = 1:NumS
```

```
extcol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);

incol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
```

```
incol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
end

%RBS settings

a1=0.625*beam_bf;
b1=0.75*beam_d;
c1=0.25*beam_bf;

Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,b1,a1);
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes);
Nodesg(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,b1,a1);
Sections(beam_sizes,NumS,extcol_sizes,incol_sizes,beam_d,beam_bf,beam_tf,be
am_tw,extcol_d,extcol_bf,extcol_tf,extcol_tw,incol_d,incol_bf,incol_tf,inco
l_tw)
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,inc
ol_bf,incol_tf,beam_d,ext_dblplate,int_dblplate)
ZeroLength(NumS)
constraints(NumS)
masses(NumS)

%SPLICES INFO

%splices coord
%number of splices and long measured from the bottom of the column in that
%story
L1=63;
L2=64.55;
L3=64.55;
%Lx=--;

%Story Splice
Sp1=3;
Sp2=5;
Sp3=7;
%Spx=--;

%If there are more than 3 splices, add de respective lines (Lx and
Spx).Then add lines in "Draw the lines of the columns" section
%If there are less,put 0 in the respective "Sp" example: (Sp2=0).

%NODES

matrix1=importdata('Nodesg.tcl',' ',0);
A=matrix1.data;
```

```
nodes=A(:,2:3);


for j=1:NumS
for i=1:size(nodes,1)
    if nodes(i,2)==h1+hi*(i-1)-beam_d(j)
       nodes(i,1)=nodes(i,1);
    else if nodes(i,2)==h1+hi*(i-1)+beam_d(j)
       nodes(i,1)=nodes(i,1);
        else if nodes(i,2)==h1+hi*(i-1)
       nodes(i,1)=nodes(i,1);
    end
        end
    end
end
end



% Plot all the nodes:
figure('Name','Structure','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on


%COLUMNS

%leaning column
for j=1:NumS
    if j==1
yc2Lc(j)=h1;
yc1Lc(j)=0;

    else
        yc2Lc(j)=h1+hi*(j-1);
        yc1Lc(j)=h1+hi*(j-2);

    end
end

%Normal columns

for j=1:NumS
    if j==1
yc2(j)=h1-beam_d(j)/2;
yc1(j)=0;

    else
        yc2(j)=h1+hi*(j-1)-beam_d(j)/2;
        yc1(j)=h1+hi*(j-2)+beam_d(j-1)/2;

    end
end
for j=1:NumS+1

    if j==1
```

```
        for i=1:3
        yf2(i)=(h1/36)*(i);

        end
        yf22{j}=yf2;
    else if j==NumS+1
            for i=1:3
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
        yf11{j}=yf1;
    else if j==2
            for i=1:3
        yf2(i)=h1+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1-(beam_d(j-1)/2)-(h1/36)*(i);
            end
            yf22{j}=yf2;
            yf11{j}=yf1;
        else
            for i=1:3
        yf2(i)=h1+hi*(j-2)+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
             yf22{j}=yf2;
             yf11{j}=yf1;
        end
        end
    end


end

a=0;
xc1=0;
xc2=0;


%Draw the lines of the columns

aa=0;
for j=1:(NumB+2)    %Number of columns including leaning column
    if j==(NumB+2)
        line([xc1+aa,xc2+aa], [yc1Lc(1),
yc2Lc(1)],'color','black','LineWidth',2.9)
    else


            line([xc1+aa,xc2+aa], [yc1(1),
yf22{1,1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,1),
yf22{1,1}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,2),
yf22{1,1}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+aa,xc2+aa], [yc2(1),
yf11{1,2}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf11{1,2}(1,1),
yf11{1,2}(1,2)],'color','black','LineWidth',2.9)
```

```
             line([xc1+aa,xc2+aa], [yf11{1,2}(1,2),
yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+aa,xc2+aa],
[yf22{1,1}(1,3),yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

    end
    aa=aa+bay;
end

for j=1:(NumB+2)
    if j==(NumB+2)
        for i=2:NumS
            line([xc1+a,xc2+a], [yc1Lc(i),
yc2Lc(i)],'color','black','LineWidth',2.9)
        end
    else

    for i=2:NumS

        if i==Sp1

            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L1],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L1,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

        else if i==Sp2
            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)
```

```
            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L2],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L2,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)



            else if i==Sp3

           line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L3],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L3,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            % add the same syntanx than before if there are more than 3
            % splices.

            % EXAMPLE:

%             else if i==Spx
%             line([xc1+a,xc2+a], [yc1(i),
yc1(i)+Lx],'color','black','LineWidth',2.9)
%             line([xc1+a,xc2+a], [yc1(i)+Lx,
yc2(i)],'color','black','LineWidth',2.9)



               else



            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),
yf11{1,i+1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),
yf11{1,i+1}(1,2)],'color','black','LineWidth',2.9)
```

```
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)


                end
            end
        end

        %close the same "if" lines added

        %EXAMPLE
        %end


    end
    end


    a=a+bay;
end

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end




%BEAMS

xob1=0;
xob2=bay;


for j=1:NumS
    for i=1:(NumB+1) % 4 bays considering leaning column
        if j==pos_splices(j)
             if i==1 %Left exterior bay
    xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;
    xb2(i)=i*xob2-incol_d(j+1)/2;
```

```
    else
        if i==NumB  %Right exterior bay
            xb1(i)=(i-1)*xob2+incol_d(j+1)/2;


            xb2(i)=i*xob2-extcol_d(j+1)/2;
        else if i==(NumB+1)
           xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;
            xb2(i)=i*xob2;
            else
                xb1(i)=(i-1)*xob2+incol_d(j+1)/2;

                xb2(i)=i*xob2-incol_d(j+1)/2;

            end
        end
    end
  else




    if i==1 %Left exterior bay
    xb1(i)=(i-1)*xob2+extcol_d(j)/2;


    xb2(i)=i*xob2-incol_d(j)/2;


    else
        if i==NumB  %Right exterior bay
            xb1(i)=(i-1)*xob2+incol_d(j)/2;


            xb2(i)=i*xob2-extcol_d(j)/2;


        else if i==(NumB+1)
           xb1(i)=(i-1)*xob2+extcol_d(j)/2;
            xb2(i)=i*xob2;
            else
                xb1(i)=(i-1)*xob2+incol_d(j)/2;

                xb2(i)=i*xob2-incol_d(j)/2;

            end
        end
    end
end
    xpi{j}=xb1;
    xpf{j}=xb2;


end
end
```

```
for i=1:NumB
for k=1:7

    if i==1

    xbf4{k}=(i-1)*xob2+(extcol_d./2)+a1+(k-1)*b1./6;

    xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;

    else if i==NumB
    xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

     xbf3{k}=i*xob2-(extcol_d./2)-a1-(k-1)*b1./6;

        else
            xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

            xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;

end

    end

xbf44{i}=xbf4;
xbf33{i}=xbf3;

end
end
```

```
yb1(1)=h1;
yb2(1)=h1;

for i=2:NumS
    yb1(i)=yb1(1)+hi*(i-1);
    yb2(i)=yb1(i);
end
```

```
% Draw the lines of the beams
redext=zeros(NumS,1);
redin=zeros(NumS,1);

for i=1:NumS

    if i==pos_splices(i)

        redext(i,1)=abs(extcol_d(i+1,1)/2-extcol_d(i,1)/2);
```

```
        redin(i,1)=abs(incol_d(i+1,1)/2-incol_d(i,1)/2);


end
end
for k=1:7
xbf44n{1,1}{1,k}=xbf44{1,1}{1,k}-redext;
xbf44n{1,2}{1,k}=xbf44{1,2}{1,k}-redin;
xbf44n{1,3}{1,k}=xbf44{1,3}{1,k}-redin;
xbf33n{1,1}{1,k}=xbf33{1,1}{1,k}+redin;
xbf33n{1,2}{1,k}=xbf33{1,2}{1,k}+redin;
xbf33n{1,3}{1,k}=xbf33{1,3}{1,k}+redext;
end

for i=1:NumS
    for j=1:NumB


        line([xpi{1,i}(1,j),xbf44n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,1}(i,1),xbf44n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,2}(i,1),xbf44n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,3}(i,1),xbf44n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,4}(i,1),xbf44n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,5}(i,1),xbf44n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,6}(i,1),xbf44n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)

        line([xpf{1,i}(1,j),xbf33n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,1}(i,1),xbf33n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,2}(i,1),xbf33n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,3}(i,1),xbf33n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,4}(i,1),xbf33n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,5}(i,1),xbf33n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,6}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        line([xbf44n{1,j}{1,7}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        end


    mm1=NumB+1;
```

```
     line([xpi{1,i}(1,mm1),xpf{1,i}(1,mm1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)



end

%PANEL ZONES

for j=1:NumS

    if j==pos_splices(j)

          for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j+1)/2;
x2p(i)=bay*(i-1)+extcol_d(j+1)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j+1)/2;
            x2p(i)=bay*(i-1)+incol_d(j+1)/2;
        end
    end


    else



    for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j)/2;
x2p(i)=bay*(i-1)+extcol_d(j)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j)/2;
            x2p(i)=bay*(i-1)+incol_d(j)/2;
        end
    end
    end
        x1pp{j}=x1p;
        x2pp{j}=x2p;
end
for i=1:(NumB+1)

x3p(i)=bay*(i-1);
end

for j=1:NumS

    yp1(j)=h1+hi*(j-1)-beam_d(j)/2;
    yp2(j)=h1+hi*(j-1)+beam_d(j)/2;
    yp3(j)=h1+hi*(j-1);
end
```

```
% Draw the lines of the panel zones

for i=1:NumS
    for j=1:(NumB+1)
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);

        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);

    end
%     k=k-1;
end

%Plot all the nodes:
figure('Name','Nodes','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on
```

ANEXO A2: CODIGO MATLAB DE EJECUCIÓN PARA MODELOS L/12 FIBER

```
clear all;
close all;
clc;

cd ('E:\maestría usfq estructuras\Tesis USFQ\modelos Lentre12\cws 5 10 5\G-
L NP 3\s02\Fiber\runner 8')

%Structural configuration:

NumS=8;     %Number of stories of the building
bay=240;    %Bay length
h1=180;     %Height of the first story
hi=156;     %Height of the rest of stories
Ry=1.1;
NumB=3;     %Number of bays without leaning column
%PROPERTIES OF THE PLASTIC HINGES:
%type=2 GRAVITY ANALYSIS!!
%type=1 REDUCTION CAPACITY OF THE COLUMN !!

type=2;

%Member sizes:

%Input the size of the member for each story from bottom to top

switch NumS
    case 2

    %TWO STORY BUILDING:

    beam_sizes = {'W30X132', 'W16X31'}';
    extcol_sizes = {'W24X131', 'W24X131'}';
    incol_sizes = {'W24X162', 'W24X162'}';

    ext_dblplate = [0.0, 0.0]';
    int_dblplate = [0.0, 0.0]';

    case 4

    %FOUR STORY BUILDING:

    beam_sizes = {'W21X73', 'W21X73', 'W21X57', 'W21X57'}';
    extcol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';
    incol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0]';
    int_dblplate = [5/16, 5/16, 5/16, 5/16]';

    case 8
    %EIGHT STORY BUILDING:

    beam_sizes = {'W30X108', 'W30X116', 'W30X116', 'W27X94',
'W27X94','W27X94','W24X84','W21X68'}';
```

```
    extcol_sizes = {'W24X131', 'W24X131', 'W24X131',
'W24X131','W24X131','W24X131','W24X131','W24X94'}';
    incol_sizes = {'W24X162', 'W24X162', 'W24X162',
'W24X162','W24X162','W24X131','W24X131','W24X94'}';


    ext_dblplate = [1/16, 1/16, 1/16, 0.0, 0.0,  0.0,  0.0,  0.0]';
    int_dblplate = [9/16, 3/8, 11/16, 3/8, 9/16, 7/16, 9/16, 5/16]';

    case 12
    %TWELVE STORY BUILDING:

    beam_sizes = {'W30X124', 'W30X132', 'W30X132', 'W30X132',
'W30X116','W30X116','W30X116','W30X116','W27X94','W27X94','W24X84','W24X84'
}';
    extcol_sizes = {'W24X207', 'W24X207', 'W24X207', 'W24X162', 'W24X162',
'W24X146', 'W24X146', 'W24X131', 'W24X131', 'W24X131', 'W24X131',
'W24X84'}';
    incol_sizes =  {'W24X207', 'W24X207', 'W24X207', 'W24X207', 'W24X207',
'W24X176', 'W24X176', 'W24X162', 'W24X162', 'W24X131', 'W24X131',
'W24X94'}';

    ext_dblplate = [0.0, 0.0, 1/16, 1/16, 0.0, 0.0, 1/16, 1/16, 0.0, 0.0,
1/16, 1/16]';
    int_dblplate = [1/2, 7/16, 5/8, 5/8, 5/8, 5/8, 11/16, 11/16, 9/16,
9/16, 9/16, 9/16]';

    case 20
    %TWENTY STORY BUILDING:

    beam_sizes = {'W33X169', 'W33X169', 'W33X169', 'W33X169', 'W33X169',
'W33X169', 'W33X169', 'W33X169', 'W33X141', 'W33X141', 'W33X141',
'W33X141', 'W33X141', 'W33X141', 'W30X108', 'W30X108', 'W30X108',
'W30X108', 'W24X62', 'W24X62'}';
    extcol_sizes = {'W14X426', 'W14X426', 'W14X426', 'W14X426', 'W14X426',
'W14X398', 'W14X398', 'W14X370', 'W14X370', 'W14X311', 'W14X311',
'W14X283', 'W14X283', 'W14X233', 'W14X233', 'W14X159', 'W14X159',
'W14X132', 'W14X132', 'W14X132'}';
    incol_sizes =  {'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X335',
'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X279', 'W24X279',
'W24X250', 'W24X250', 'W24X250', 'W24X250', 'W24X162', 'W24X162',
'W24X162', 'W24X162', 'W24X103'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1/16, 1/16, 1/4, 1/4, 3/8, 3/8, 0.0, 0.0]';
    int_dblplate = [1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 5/16, 5/16,
1/2, 1/2, 1/2, 1/2, 9/16, 9/16, 9/16, 9/16, 3/16, 3/16 ]';

end

%Material properties:

Fy = 55; % Nominal yield point (ksi)
E_steel = 29000; % ksi
nu_steel = 0.3;
G_steel = E_steel/(2*(1+nu_steel)); % ksi
```

```
%alpha = 0.03; % hardening in third segment of trilinear panel zone
material

%Read AISC.text file which contain mechanical properties of the cross
sections

AISC_ID = fopen('AISC.txt');
AISC_data = textscan(AISC_ID, '%s %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f');

beam_A      = zeros(NumS,1); % area
beam_d      = zeros(NumS,1); % depth
beam_bf     = zeros(NumS,1); % flange length
beam_tw     = zeros(NumS,1); % web thickness
beam_tf     = zeros(NumS,1); % flange thickness
beam_bf_2tf = zeros(NumS,1); % bf/2tf
beam_h_tw   = zeros(NumS,1); % h/tw
beam_Ix     = zeros(NumS,1);
beam_Zx     = zeros(NumS,1);
beam_Sx     = zeros(NumS,1);
beam_rx     = zeros(NumS,1);
beam_Iy     = zeros(NumS,1);
beam_Zy     = zeros(NumS,1);
beam_Sy     = zeros(NumS,1);
beam_ry     = zeros(NumS,1);
beam_J      = zeros(NumS,1);
beam_Aweb   = zeros(NumS,1); % equivalent to (d-2tf)*tw

extcol_A      = zeros(NumS,1);
extcol_d      = zeros(NumS,1);
extcol_bf     = zeros(NumS,1);
extcol_tw     = zeros(NumS,1);
extcol_tf     = zeros(NumS,1);
extcol_bf_2tf = zeros(NumS,1);
extcol_h_tw   = zeros(NumS,1);
extcol_Ix     = zeros(NumS,1);
extcol_Zx     = zeros(NumS,1);
extcol_Sx     = zeros(NumS,1);
extcol_rx     = zeros(NumS,1);
extcol_Iy     = zeros(NumS,1);
extcol_Zy     = zeros(NumS,1);
extcol_Sy     = zeros(NumS,1);
extcol_ry     = zeros(NumS,1);
extcol_J      = zeros(NumS,1);
extcol_Aweb   = zeros(NumS,1);

incol_A      = zeros(NumS,1);
incol_d      = zeros(NumS,1);
incol_bf     = zeros(NumS,1);
incol_tw     = zeros(NumS,1);
incol_tf     = zeros(NumS,1);
incol_bf_2tf = zeros(NumS,1);
incol_h_tw   = zeros(NumS,1);
incol_Ix     = zeros(NumS,1);
incol_Zx     = zeros(NumS,1);
incol_Sx     = zeros(NumS,1);
incol_rx     = zeros(NumS,1);
incol_Iy     = zeros(NumS,1);
```

```
incol_Zy     = zeros(NumS,1);
incol_Sy     = zeros(NumS,1);
incol_ry     = zeros(NumS,1);
incol_J      = zeros(NumS,1);
incol_Aweb   = zeros(NumS,1);

%Load cross sections properties of each story

for i = 1:NumS

beam_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);

end

for i = 1:NumS

extcol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
```

```
extcol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);

incol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
```

```
incol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
end

%RBS settings

a1=0.625*beam_bf;
b1=0.75*beam_d;
c1=0.25*beam_bf;
%
% %Plastic Moment at RBS location
%
% RBS_Zx=beam_Zx-2*c.*beam_tf.*(beam_d-beam_tf);
%
% %Max. Probable Moment at RBS
%
% My_RBS=Ry*RBS_Zx*Fy;
%
% sh_ext=a+b/2+extcol_d/2;
% sh_int=a+b/2+incol_d/2;
%
% %Beam Lengths:
%
% beam_length_ext=bay-sh_ext-sh_int;
% beam_length_int=bay-2*sh_int;
%
%
% %Column Lengths:
%
% h_first=h1-beam_d(1,1);
%
% htypical=ones(NumS-1,1)*hi;
% h_i(1)=h_first;
%
%
% for i=2:NumS
%
%  % h_i(i)=htypical(i-1)-beam_d(i-1,1)/2-beam_d(i,1)/2;
%  h_i(i)=htypical(i-1)-beam_d(i-1);
%
% end
%
%
nf=10;   %%This value of 10 comes from Ibarra's Dissertation. Pag 299.
% I=ones(1,NumS);
% Mp_My = 0.1;
% Mc_My=1.1;
% Lb=150;
%
%
% if type==1
%     [factor_ext,factor_int]=reduction_capacity(NumS,extcol_A,incol_A,Fy);
%     reduction_capacity_ext=factor_ext;
%     reduction_capacity_int=factor_int;
% else
%
%     reduction_capacity_ext=ones(1,NumS);
%     reduction_capacity_int=ones(1,NumS);
% end
```

```
%
% for i=1:NumS
%
%    Vy_ext(i)=2*(Ry*Fy*RBS_Zx(i)*1.15)/beam_length_ext(i)+(3.0274e-
01)*beam_length_ext(i)/2;
%
% end
%
% Ptotal=sum(Vy_ext);
%
% for i=1:NumS
%
%    Vy_int(i)=2*(Ry*Fy*incol_Zx(i))/beam_length_int(i)+2.9481e-
01*beam_length_int(i)/2;
%
% end

Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,b1,a1);
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes);
Nodesg(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,b1,a1);
Sections(beam_sizes,NumS,extcol_sizes,incol_sizes,beam_d,beam_bf,beam_tf,be
am_tw,extcol_d,extcol_bf,extcol_tf,extcol_tw,incol_d,incol_bf,incol_tf,inco
l_tw)
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,inc
ol_bf,incol_tf,beam_d,ext_dblplate,int_dblplate)
ZeroLength(NumS)
constraints(NumS)
masses(NumS)









%SPLICES INFO

%splices coord
%number of splices and long measured from the bottom of the column in that
%story
L1=63;
L2=64.55;
L3=64.55;
%Lx=--;

%Story Splice
Sp1=3;
Sp2=5;
Sp3=7;
%Spx=--;

%If there are more than 3 splices, add de respective lines (Lx and
Spx).Then add lines in "Draw the lines of the columns" section
%If there are less,put 0 in the respective "Sp" example: (Sp2=0).
```

```
%NODES

matrix1=importdata('Nodesg.tcl',' ',0);
A=matrix1.data;
nodes=A(:,2:3);


for j=1:NumS
for i=1:size(nodes,1)
    if nodes(i,2)==h1+hi*(i-1)-beam_d(j)
       nodes(i,1)=nodes(i,1);
    else if nodes(i,2)==h1+hi*(i-1)+beam_d(j)
       nodes(i,1)=nodes(i,1);
        else if nodes(i,2)==h1+hi*(i-1)
       nodes(i,1)=nodes(i,1);
    end
        end
    end
end
end



% Plot all the nodes:
figure('Name','Structure','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on


%COLUMNS

%leaning column
for j=1:NumS
    if j==1
yc2Lc(j)=h1;
yc1Lc(j)=0;

    else
        yc2Lc(j)=h1+hi*(j-1);
        yc1Lc(j)=h1+hi*(j-2);

    end
end

%Normal columns

for j=1:NumS
    if j==1
yc2(j)=h1-beam_d(j)/2;
yc1(j)=0;

    else
        yc2(j)=h1+hi*(j-1)-beam_d(j)/2;
        yc1(j)=h1+hi*(j-2)+beam_d(j-1)/2;
```

```
    end
end
for j=1:NumS+1

    if j==1
        for i=1:3
        yf2(i)=(h1/36)*(i);

        end
        yf22{j}=yf2;
    else if j==NumS+1
            for i=1:3
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
        yf11{j}=yf1;
    else if j==2
            for i=1:3
        yf2(i)=h1+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1-(beam_d(j-1)/2)-(h1/36)*(i);
            end
            yf22{j}=yf2;
            yf11{j}=yf1;
        else
            for i=1:3
        yf2(i)=h1+hi*(j-2)+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
             yf22{j}=yf2;
             yf11{j}=yf1;
        end
        end
    end


end

a=0;
xc1=0;
xc2=0;


%Draw the lines of the columns

aa=0;
for j=1:(NumB+2)    %Number of columns including leaning column
    if j==(NumB+2)
        line([xc1+aa,xc2+aa], [yc1Lc(1),
yc2Lc(1)],'color','black','LineWidth',2.9)
    else


            line([xc1+aa,xc2+aa], [yc1(1),
yf22{1,1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,1),
yf22{1,1}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,2),
yf22{1,1}(1,3)],'color','black','LineWidth',2.9)
```

```
            line([xc1+aa,xc2+aa], [yc2(1),
yf11{1,2}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf11{1,2}(1,1),
yf11{1,2}(1,2)],'color','black','LineWidth',2.9)
             line([xc1+aa,xc2+aa], [yf11{1,2}(1,2),
yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+aa,xc2+aa],
[yf22{1,1}(1,3),yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

    end
    aa=aa+bay;
end

for j=1:(NumB+2)
    if j==(NumB+2)
        for i=2:NumS
            line([xc1+a,xc2+a], [yc1Lc(i),
yc2Lc(i)],'color','black','LineWidth',2.9)
        end
    else

    for i=2:NumS

        if i==Sp1

            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L1],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L1,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

        else if i==Sp2
            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
```

```
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L2],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L2,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)




            else if i==Sp3

           line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L3],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L3,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            % add the same syntanx than before if there are more than 3
            % splices.

            % EXAMPLE:

%             else if i==Spx
%             line([xc1+a,xc2+a], [yc1(i),
yc1(i)+Lx],'color','black','LineWidth',2.9)
%             line([xc1+a,xc2+a], [yc1(i)+Lx,
yc2(i)],'color','black','LineWidth',2.9)



               else


            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
```

```
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),
yf11{1,i+1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),
yf11{1,i+1}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)


                end
            end
        end

        %close the same "if" lines added

        %EXAMPLE
        %end


    end
    end


    a=a+bay;
end

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end




%BEAMS

xob1=0;
xob2=bay;


for j=1:NumS
    for i=1:(NumB+1) % 4 bays considering leaning column
```

```
    if j==pos_splices(j)
         if i==1 %Left exterior bay
xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;



xb2(i)=i*xob2-incol_d(j+1)/2;




else
    if i==NumB  %Right exterior bay
        xb1(i)=(i-1)*xob2+incol_d(j+1)/2;




        xb2(i)=i*xob2-extcol_d(j+1)/2;




    else if i==(NumB+1)
       xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;
        xb2(i)=i*xob2;
        else
            xb1(i)=(i-1)*xob2+incol_d(j+1)/2;




            xb2(i)=i*xob2-incol_d(j+1)/2;



        end
    end
end
```

```
else


  if i==1 %Left exterior bay
  xb1(i)=(i-1)*xob2+extcol_d(j)/2;



  xb2(i)=i*xob2-incol_d(j)/2;



  else
      if i==NumB  %Right exterior bay
          xb1(i)=(i-1)*xob2+incol_d(j)/2;



          xb2(i)=i*xob2-extcol_d(j)/2;



      else if i==(NumB+1)
         xb1(i)=(i-1)*xob2+extcol_d(j)/2;
          xb2(i)=i*xob2;
          else
              xb1(i)=(i-1)*xob2+incol_d(j)/2;



              xb2(i)=i*xob2-incol_d(j)/2;
```

```
            end
        end
    end
end
    xpi{j}=xb1;
    xpf{j}=xb2;


end
end
```

```
for i=1:NumB
for k=1:7

    if i==1

    xbf4{k}=(i-1)*xob2+(extcol_d./2)+a1+(k-1)*b1./6;

    xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;

    else if i==NumB
    xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

     xbf3{k}=i*xob2-(extcol_d./2)-a1-(k-1)*b1./6;

        else
            xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

            xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;



end



    end



xbf44{i}=xbf4;
xbf33{i}=xbf3;



end
```

```
end




yb1(1)=h1;
yb2(1)=h1;

for i=2:NumS
    yb1(i)=yb1(1)+hi*(i-1);
    yb2(i)=yb1(i);
end



% Draw the lines of the beams
redext=zeros(NumS,1);
redin=zeros(NumS,1);

for i=1:NumS

    if i==pos_splices(i)

        redext(i,1)=abs(extcol_d(i+1,1)/2-extcol_d(i,1)/2);
        redin(i,1)=abs(incol_d(i+1,1)/2-incol_d(i,1)/2);


end
end
for k=1:7
xbf44n{1,1}{1,k}=xbf44{1,1}{1,k}-redext;
xbf44n{1,2}{1,k}=xbf44{1,2}{1,k}-redin;
xbf44n{1,3}{1,k}=xbf44{1,3}{1,k}-redin;
xbf33n{1,1}{1,k}=xbf33{1,1}{1,k}+redin;
xbf33n{1,2}{1,k}=xbf33{1,2}{1,k}+redin;
xbf33n{1,3}{1,k}=xbf33{1,3}{1,k}+redext;
end

for i=1:NumS
    for j=1:NumB


        line([xpi{1,i}(1,j),xbf44n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,1}(i,1),xbf44n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,2}(i,1),xbf44n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,3}(i,1),xbf44n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,4}(i,1),xbf44n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
```

```
        line([xbf44n{1,j}{1,5}(i,1),xbf44n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,6}(i,1),xbf44n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)

        line([xpf{1,i}(1,j),xbf33n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,1}(i,1),xbf33n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,2}(i,1),xbf33n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,3}(i,1),xbf33n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,4}(i,1),xbf33n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,5}(i,1),xbf33n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,6}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        line([xbf44n{1,j}{1,7}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        end









    mm1=NumB+1;

     line([xpi{1,i}(1,mm1),xpf{1,i}(1,mm1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)



end









%PANEL ZONES

for j=1:NumS
```

```
    if j==pos_splices(j)

          for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j+1)/2;
x2p(i)=bay*(i-1)+extcol_d(j+1)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j+1)/2;
            x2p(i)=bay*(i-1)+incol_d(j+1)/2;
        end
    end


    else



    for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j)/2;
x2p(i)=bay*(i-1)+extcol_d(j)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j)/2;
            x2p(i)=bay*(i-1)+incol_d(j)/2;
        end
    end
    end
        x1pp{j}=x1p;
        x2pp{j}=x2p;
end
for i=1:(NumB+1)

x3p(i)=bay*(i-1);
end

for j=1:NumS

    yp1(j)=h1+hi*(j-1)-beam_d(j)/2;
    yp2(j)=h1+hi*(j-1)+beam_d(j)/2;
    yp3(j)=h1+hi*(j-1);
end



% Draw the lines of the panel zones

for i=1:NumS
    for j=1:(NumB+1)
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);
```

```
        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);

    end
%     k=k-1;
end

%Plot all the nodes:
figure('Name','Nodes','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on
```

ANEXO A3: CODIGO MATLAB DE EJECUCIÓN PARA MODELOS L/12 FIBER HINGE AND CPH

```
clear all;
close all;
clc;

cd ('E:\maestría usfq estructuras\Tesis USFQ\modelos Lentre12\cws 5 10 5\G-
L NP 3\s02\Fiber Hinge and CPH\runner 8')

%Structural configuration:

NumS=8;     %Number of stories of the building
bay=240;    %Bay length
h1=180;     %Height of the first story
hi=156;     %Height of the rest of stories
Ry=1.1;
NumB=3;     %Number of bays without leaning column
%PROPERTIES OF THE PLASTIC HINGES:
%type=2 GRAVITY ANALYSIS!!
%type=1 REDUCTION CAPACITY OF THE COLUMN !!

type=2;

%Member sizes:

%Input the size of the member for each story from bottom to top

switch NumS
    case 2

    %TWO STORY BUILDING:

    beam_sizes = {'W30X132', 'W16X31'}';
    extcol_sizes = {'W24X131', 'W24X131'}';
    incol_sizes = {'W24X162', 'W24X162'}';

    ext_dblplate = [0.0, 0.0]';
    int_dblplate = [0.0, 0.0]';

    case 4

    %FOUR STORY BUILDING:

    beam_sizes = {'W21X73', 'W21X73', 'W21X57', 'W21X57'}';
    extcol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';
    incol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0]';
    int_dblplate = [5/16, 5/16, 5/16, 5/16]';

    case 8
    %EIGHT STORY BUILDING:
```

```
    beam_sizes = {'W30X108', 'W30X116', 'W30X116', 'W27X94',
'W27X94','W27X94','W24X84','W21X68'}';
    extcol_sizes = {'W24X131', 'W24X131', 'W24X131',
'W24X131','W24X131','W24X131','W24X131','W24X94'}';
    incol_sizes = {'W24X162', 'W24X162', 'W24X162',
'W24X162','W24X162','W24X131','W24X131','W24X94'}';


    ext_dblplate = [1/16, 1/16, 1/16, 0.0, 0.0,  0.0,  0.0,  0.0]';
    int_dblplate = [9/16, 3/8, 11/16, 3/8, 9/16, 7/16, 9/16, 5/16]';

    case 12
    %TWELVE STORY BUILDING:

    beam_sizes = {'W30X124', 'W30X132', 'W30X132', 'W30X132',
'W30X116','W30X116','W30X116','W30X116','W27X94','W27X94','W24X84','W24X84'
}';
    extcol_sizes = {'W24X207', 'W24X207', 'W24X207', 'W24X162', 'W24X162',
'W24X146', 'W24X146', 'W24X131', 'W24X131', 'W24X131', 'W24X131',
'W24X84'}';
    incol_sizes =  {'W24X207', 'W24X207', 'W24X207', 'W24X207', 'W24X207',
'W24X176', 'W24X176', 'W24X162', 'W24X162', 'W24X131', 'W24X131',
'W24X94'}';

    ext_dblplate = [0.0, 0.0, 1/16, 1/16, 0.0, 0.0, 1/16, 1/16, 0.0, 0.0,
1/16, 1/16]';
    int_dblplate = [1/2, 7/16, 5/8, 5/8, 5/8, 5/8, 11/16, 11/16, 9/16,
9/16, 9/16, 9/16]';

    case 20
    %TWENTY STORY BUILDING:

    beam_sizes = {'W33X169', 'W33X169', 'W33X169', 'W33X169', 'W33X169',
'W33X169', 'W33X169', 'W33X169', 'W33X141', 'W33X141', 'W33X141',
'W33X141', 'W33X141', 'W33X141', 'W30X108', 'W30X108', 'W30X108',
'W30X108', 'W24X62', 'W24X62'}';
    extcol_sizes = {'W14X426', 'W14X426', 'W14X426', 'W14X426', 'W14X426',
'W14X398', 'W14X398', 'W14X370', 'W14X370', 'W14X311', 'W14X311',
'W14X283', 'W14X283', 'W14X233', 'W14X233', 'W14X159', 'W14X159',
'W14X132', 'W14X132', 'W14X132'}';
    incol_sizes =  {'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X335',
'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X279', 'W24X279',
'W24X250', 'W24X250', 'W24X250', 'W24X250', 'W24X162', 'W24X162',
'W24X162', 'W24X162', 'W24X103'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1/16, 1/16, 1/4, 1/4, 3/8, 3/8, 0.0, 0.0]';
    int_dblplate = [1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 5/16, 5/16,
1/2, 1/2, 1/2, 1/2, 9/16, 9/16, 9/16, 9/16, 3/16, 3/16 ]';

end

%Material properties:

Fy = 55; % Nominal yield point (ksi)
E_steel = 29000; % ksi
nu_steel = 0.3;
```

```
G_steel = E_steel/(2*(1+nu_steel)); % ksi
%alpha = 0.03; % hardening in third segment of trilinear panel zone
material

%Read AISC.text file which contain mechanical properties of the cross
sections

AISC_ID = fopen('AISC.txt');
AISC_data = textscan(AISC_ID, '%s %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f');

beam_A      = zeros(NumS,1); % area
beam_d      = zeros(NumS,1); % depth
beam_bf     = zeros(NumS,1); % flange length
beam_tw     = zeros(NumS,1); % web thickness
beam_tf     = zeros(NumS,1); % flange thickness
beam_bf_2tf = zeros(NumS,1); % bf/2tf
beam_h_tw   = zeros(NumS,1); % h/tw
beam_Ix     = zeros(NumS,1);
beam_Zx     = zeros(NumS,1);
beam_Sx     = zeros(NumS,1);
beam_rx     = zeros(NumS,1);
beam_Iy     = zeros(NumS,1);
beam_Zy     = zeros(NumS,1);
beam_Sy     = zeros(NumS,1);
beam_ry     = zeros(NumS,1);
beam_J      = zeros(NumS,1);
beam_Aweb   = zeros(NumS,1); % equivalent to (d-2tf)*tw

extcol_A      = zeros(NumS,1);
extcol_d      = zeros(NumS,1);
extcol_bf     = zeros(NumS,1);
extcol_tw     = zeros(NumS,1);
extcol_tf     = zeros(NumS,1);
extcol_bf_2tf = zeros(NumS,1);
extcol_h_tw   = zeros(NumS,1);
extcol_Ix     = zeros(NumS,1);
extcol_Zx     = zeros(NumS,1);
extcol_Sx     = zeros(NumS,1);
extcol_rx     = zeros(NumS,1);
extcol_Iy     = zeros(NumS,1);
extcol_Zy     = zeros(NumS,1);
extcol_Sy     = zeros(NumS,1);
extcol_ry     = zeros(NumS,1);
extcol_J      = zeros(NumS,1);
extcol_Aweb   = zeros(NumS,1);

incol_A      = zeros(NumS,1);
incol_d      = zeros(NumS,1);
incol_bf     = zeros(NumS,1);
incol_tw     = zeros(NumS,1);
incol_tf     = zeros(NumS,1);
incol_bf_2tf = zeros(NumS,1);
incol_h_tw   = zeros(NumS,1);
incol_Ix     = zeros(NumS,1);
incol_Zx     = zeros(NumS,1);
incol_Sx     = zeros(NumS,1);
incol_rx     = zeros(NumS,1);
```

```
incol_Iy     = zeros(NumS,1);
incol_Zy     = zeros(NumS,1);
incol_Sy     = zeros(NumS,1);
incol_ry     = zeros(NumS,1);
incol_J      = zeros(NumS,1);
incol_Aweb   = zeros(NumS,1);

%Load cross sections properties of each story

for i = 1:NumS

beam_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);

end

for i = 1:NumS

extcol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
```

```
extcol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);

incol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
```

```
incol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
end

%RBS settings

a1=0.625*beam_bf;
b1=0.75*beam_d;
c1=0.25*beam_bf;

a=0.625*beam_bf;
b=0.75*beam_d;
c=0.25*beam_bf;
%
%Plastic Moment at RBS location

RBS_Zx=beam_Zx-2*c.*beam_tf.*(beam_d-beam_tf);

%Max. Probable Moment at RBS

My_RBS=Ry*RBS_Zx*Fy;

sh_ext=a+b/2+extcol_d/2;
sh_int=a+b/2+incol_d/2;

%Beam Lengths:

beam_length_ext=bay-sh_ext-sh_int;
beam_length_int=bay-2*sh_int;


%Column Lengths:

h_first=h1-beam_d(1,1);

htypical=ones(NumS-1,1)*hi;
h_i(1)=h_first;


for i=2:NumS

 % h_i(i)=htypical(i-1)-beam_d(i-1,1)/2-beam_d(i,1)/2;
 h_i(i)=htypical(i-1)-beam_d(i-1);

end


nf=10;   %%This value of 10 comes from Ibarra's Dissertation. Pag 299.
I=ones(1,NumS);
Mp_My = 0.1;
Mc_My=1.1;
Lb=150;


if type==1
```

```
    [factor_ext,factor_int]=reduction_capacity(NumS,extcol_A,incol_A,Fy);
    reduction_capacity_ext=factor_ext;
    reduction_capacity_int=factor_int;
else

    reduction_capacity_ext=ones(1,NumS);
    reduction_capacity_int=ones(1,NumS);
end

for i=1:NumS

   Vy_ext(i)=2*(Ry*Fy*RBS_Zx(i)*1.15)/beam_length_ext(i)+(3.0274e-
01)*beam_length_ext(i)/2;

end

Ptotal=sum(Vy_ext);

for i=1:NumS

   Vy_int(i)=2*(Ry*Fy*incol_Zx(i))/beam_length_int(i)+2.9481e-
01*beam_length_int(i)/2;

end

for i=1:NumS


    %Spring properties of the columns (Non-RBS connections):
    %Exterior columns:

    ext_col_theta_p(i) =  0.087*(extcol_h_tw(i)^(-
0.365))*(extcol_bf_2tf(i)^(-
0.14))*((Lb/extcol_d(i))^(0.34))*(extcol_d(i)/21)^(-0.721)*(Fy/50)^(-0.23);
    ext_col_theta_pc(i) = 5.7*extcol_h_tw(i)^(-0.565)*extcol_bf_2tf(i)^(-
0.80)*(extcol_d(i)/21)^(-0.28)*(Fy/50)^(-0.43);
    ext_col_lambda(i) =   500*extcol_h_tw(i)^(-1.34)*extcol_bf_2tf(i)^(-
0.595)*(Fy/50)^(-0.36);

    Ke_ext_col(i) = (nf+1)*6.0*E_steel*extcol_Ix(i)/h_i(i);
    My_ext_col(i) = Ry*Fy*extcol_Zx(i);
    Kp_ext(i)=(Mc_My*My_ext_col(i)-My_ext_col(i))/(ext_col_theta_p(i));
    ass_ext_col(i) =Kp_ext(i)/Ke_ext_col(i);

    %Interior columns:

    int_col_theta_p(i) =  0.087*(incol_h_tw(i)^(-
0.365))*(incol_bf_2tf(i)^(-
0.14))*(Lb/(incol_d(i)))^(0.34)*(incol_d(i)/21)^(-0.721)*(Fy/50)^(-0.23);
    int_col_theta_pc(i) = 5.7*(incol_h_tw(i)^(-0.565))*(incol_bf_2tf(i)^(-
0.80))*(incol_d(i)/21)^(-0.28)*(Fy/50)^(-0.43);
    int_col_lambda(i) =   500*(incol_h_tw(i)^(-1.34))*(incol_bf_2tf(i)^(-
0.595))*(Fy/50)^(-0.36);

    Ke_int_col(i) = (nf+1)*6.0*E_steel*incol_Ix(i)/h_i(i);
    My_int_col(i) = Ry*Fy*incol_Zx(i);  %*red_fac;
```

```
    Kp_int(i)=(Mc_My*My_int_col(i)-My_int_col(i))/(int_col_theta_p(i));
    ass_int_col(i) =Kp_int(i)/Ke_int_col(i);


    % Spring properties of the beams (RBS connections):

    beam_theta_p(i) =  0.19*(beam_h_tw(i)^(-0.314))*(beam_bf_2tf(i)^(-
0.10))*(50)^(-0.1185) *(Lb/beam_d(i))^(0.113)*(beam_d(i)/21)^(-
0.76)*(Fy/50)^(-0.07);
    beam_theta_pc(i) = 9.62*beam_h_tw(i)^(-0.513)*beam_bf_2tf(i)^(-
0.863)*(50)^(-0.108) *(Fy/50)^(-0.36);
    beam_lambda(i) =   592*beam_h_tw(i)^(-1.138) *beam_bf_2tf(i)^(-
0.632)*(50)^(-0.205) *(Fy/50)^(-0.391);

    Ke_beam(i) = (nf+1)*6.0*E_steel*beam_Ix(i)*0.9/beam_length_int(i);
    Kp(i)=(Mc_My*My_RBS(i)-My_RBS(i))/(beam_theta_p(i));
    ass_beam(i) =Kp(i)/Ke_beam(i);


end

Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,sh_int,sh_ext);
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes,nf);
Nodesg(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,sh_int,sh_ext);
Sections(beam_sizes,NumS,extcol_sizes,incol_sizes,beam_d,beam_bf,beam_tf,be
am_tw,extcol_d,extcol_bf,extcol_tf,extcol_tw,incol_d,incol_bf,incol_tf,inco
l_tw)
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,inc
ol_bf,incol_tf,beam_d,ext_dblplate,int_dblplate,Ke_beam,ass_beam,My_RBS,bea
m_lambda,beam_theta_p,beam_theta_pc)
ZeroLength(NumS)
constraints(NumS)
masses(NumS)




%SPLICES INFO

%splices coord
%number of splices and long measured from the bottom of the column in that
%story
L1=63;
L2=64.55;
L3=64.55;
%Lx=--;

%Story Splice
Sp1=3;
Sp2=5;
Sp3=7;
```

```
%Spx=--;

%If there are more than 3 splices, add de respective lines (Lx and
Spx).Then add lines in "Draw the lines of the columns" section
%If there are less,put 0 in the respective "Sp" example: (Sp2=0).

%NODES

matrix1=importdata('Nodesg.tcl',' ',0);
A=matrix1.data;
nodes=A(:,2:3);


for j=1:NumS
for i=1:size(nodes,1)
    if nodes(i,2)==h1+hi*(i-1)-beam_d(j)
       nodes(i,1)=nodes(i,1);
    else if nodes(i,2)==h1+hi*(i-1)+beam_d(j)
       nodes(i,1)=nodes(i,1);
        else if nodes(i,2)==h1+hi*(i-1)
       nodes(i,1)=nodes(i,1);
    end
        end
    end
end
end



% Plot all the nodes:
figure('Name','Structure','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on


%COLUMNS

%leaning column
for j=1:NumS
    if j==1
yc2Lc(j)=h1;
yc1Lc(j)=0;

    else
        yc2Lc(j)=h1+hi*(j-1);
        yc1Lc(j)=h1+hi*(j-2);

    end
end

%Normal columns

for j=1:NumS
    if j==1
yc2(j)=h1-beam_d(j)/2;
yc1(j)=0;
```

```
    else
        yc2(j)=h1+hi*(j-1)-beam_d(j)/2;
        yc1(j)=h1+hi*(j-2)+beam_d(j-1)/2;

    end
end
for j=1:NumS+1

    if j==1
        for i=1:3
        yf2(i)=(h1/36)*(i);

        end
        yf22{j}=yf2;
    else if j==NumS+1
            for i=1:3
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
        yf11{j}=yf1;
    else if j==2
            for i=1:3
        yf2(i)=h1+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1-(beam_d(j-1)/2)-(h1/36)*(i);
            end
            yf22{j}=yf2;
            yf11{j}=yf1;
        else
            for i=1:3
        yf2(i)=h1+hi*(j-2)+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
             yf22{j}=yf2;
             yf11{j}=yf1;
        end
        end
    end


end

a=0;
xc1=0;
xc2=0;


%Draw the lines of the columns

aa=0;
for j=1:(NumB+2)    %Number of columns including leaning column
    if j==(NumB+2)
        line([xc1+aa,xc2+aa], [yc1Lc(1),
yc2Lc(1)],'color','black','LineWidth',2.9)
    else
```

```
            line([xc1+aa,xc2+aa], [yc1(1),
yf22{1,1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,1),
yf22{1,1}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,2),
yf22{1,1}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+aa,xc2+aa], [yc2(1),
yf11{1,2}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf11{1,2}(1,1),
yf11{1,2}(1,2)],'color','black','LineWidth',2.9)
             line([xc1+aa,xc2+aa], [yf11{1,2}(1,2),
yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+aa,xc2+aa],
[yf22{1,1}(1,3),yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

    end
    aa=aa+bay;
end

for j=1:(NumB+2)
    if j==(NumB+2)
        for i=2:NumS
            line([xc1+a,xc2+a], [yc1Lc(i),
yc2Lc(i)],'color','black','LineWidth',2.9)
        end
    else

    for i=2:NumS

        if i==Sp1

            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L1],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L1,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

        else if i==Sp2
            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
```

```
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L2],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L2,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)




            else if i==Sp3

           line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L3],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L3,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            % add the same syntanx than before if there are more than 3
            % splices.

            % EXAMPLE:

%             else if i==Spx
%             line([xc1+a,xc2+a], [yc1(i),
yc1(i)+Lx],'color','black','LineWidth',2.9)
%             line([xc1+a,xc2+a], [yc1(i)+Lx,
yc2(i)],'color','black','LineWidth',2.9)



                else
```

```
            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),
yf11{1,i+1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),
yf11{1,i+1}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)


                end
            end
        end

        %close the same "if" lines added

        %EXAMPLE
        %end


    end
    end


    a=a+bay;
end

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end




%BEAMS

xob1=0;
```

```
xob2=bay;


for j=1:NumS
    for i=1:(NumB+1) % 4 bays considering leaning column
        if j==pos_splices(j)
             if i==1 %Left exterior bay
    xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;




    xb2(i)=i*xob2-incol_d(j+1)/2;






    else
        if i==NumB  %Right exterior bay
            xb1(i)=(i-1)*xob2+incol_d(j+1)/2;




            xb2(i)=i*xob2-extcol_d(j+1)/2;




        else if i==(NumB+1)
           xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;
            xb2(i)=i*xob2;
            else
                xb1(i)=(i-1)*xob2+incol_d(j+1)/2;




                xb2(i)=i*xob2-incol_d(j+1)/2;



            end
```

```
        end
    end
```

```
else
```

```
    if i==1 %Left exterior bay
    xb1(i)=(i-1)*xob2+extcol_d(j)/2;
```

```
    xb2(i)=i*xob2-incol_d(j)/2;
```

```
    else
        if i==NumB  %Right exterior bay
            xb1(i)=(i-1)*xob2+incol_d(j)/2;
```

```
            xb2(i)=i*xob2-extcol_d(j)/2;
```

```
        else if i==(NumB+1)
           xb1(i)=(i-1)*xob2+extcol_d(j)/2;
            xb2(i)=i*xob2;
            else
                xb1(i)=(i-1)*xob2+incol_d(j)/2;
```

```
                xb2(i)=i*xob2-incol_d(j)/2;



            end
        end
    end
end
    xpi{j}=xb1;
    xpf{j}=xb2;


end
end




for i=1:NumB
for k=1:7

    if i==1

    xbf4{k}=(i-1)*xob2+(extcol_d./2)+a1+(k-1)*b1./6;

    xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;

    else if i==NumB
    xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

     xbf3{k}=i*xob2-(extcol_d./2)-a1-(k-1)*b1./6;

        else
            xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

            xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;



end



    end



xbf44{i}=xbf4;
xbf33{i}=xbf3;
```

```
end
end




yb1(1)=h1;
yb2(1)=h1;

for i=2:NumS
    yb1(i)=yb1(1)+hi*(i-1);
    yb2(i)=yb1(i);
end



% Draw the lines of the beams
redext=zeros(NumS,1);
redin=zeros(NumS,1);

for i=1:NumS

    if i==pos_splices(i)

        redext(i,1)=abs(extcol_d(i+1,1)/2-extcol_d(i,1)/2);
        redin(i,1)=abs(incol_d(i+1,1)/2-incol_d(i,1)/2);


end
end
for k=1:7
xbf44n{1,1}{1,k}=xbf44{1,1}{1,k}-redext;
xbf44n{1,2}{1,k}=xbf44{1,2}{1,k}-redin;
xbf44n{1,3}{1,k}=xbf44{1,3}{1,k}-redin;
xbf33n{1,1}{1,k}=xbf33{1,1}{1,k}+redin;
xbf33n{1,2}{1,k}=xbf33{1,2}{1,k}+redin;
xbf33n{1,3}{1,k}=xbf33{1,3}{1,k}+redext;
end

for i=1:NumS
    for j=1:NumB


        line([xpi{1,i}(1,j),xbf44n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,1}(i,1),xbf44n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,2}(i,1),xbf44n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
```

```
        line([xbf44n{1,j}{1,3}(i,1),xbf44n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,4}(i,1),xbf44n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,5}(i,1),xbf44n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,6}(i,1),xbf44n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)

        line([xpf{1,i}(1,j),xbf33n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,1}(i,1),xbf33n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,2}(i,1),xbf33n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,3}(i,1),xbf33n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,4}(i,1),xbf33n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,5}(i,1),xbf33n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,6}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        line([xbf44n{1,j}{1,7}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        end









    mm1=NumB+1;

     line([xpi{1,i}(1,mm1),xpf{1,i}(1,mm1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)



end
```

```
%PANEL ZONES

for j=1:NumS

    if j==pos_splices(j)

          for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j+1)/2;
x2p(i)=bay*(i-1)+extcol_d(j+1)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j+1)/2;
            x2p(i)=bay*(i-1)+incol_d(j+1)/2;
        end
    end


    else



    for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j)/2;
x2p(i)=bay*(i-1)+extcol_d(j)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j)/2;
            x2p(i)=bay*(i-1)+incol_d(j)/2;
        end
    end
    end
        x1pp{j}=x1p;
        x2pp{j}=x2p;
end
for i=1:(NumB+1)

x3p(i)=bay*(i-1);
end

for j=1:NumS

    yp1(j)=h1+hi*(j-1)-beam_d(j)/2;
    yp2(j)=h1+hi*(j-1)+beam_d(j)/2;
    yp3(j)=h1+hi*(j-1);
end



% Draw the lines of the panel zones

for i=1:NumS
    for j=1:(NumB+1)
```

```
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);

        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);

    end
%     k=k-1;
end

%Plot all the nodes:
figure('Name','Nodes','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on
```

ANEXO A4: CODIGO MATLAB DE EJECUCIÓN PARA MODELOS L/12 FIBER AND CPH

```
clear all;
close all;
clc;

cd ('E:\maestría usfq estructuras\Tesis USFQ\modelos Lentre12\cws 5 10 5\G-
L NP 3\s02\Fiber and CPH\runner 8')

%Structural configuration:

NumS=8;     %Number of stories of the building
bay=240;    %Bay length
h1=180;     %Height of the first story
hi=156;     %Height of the rest of stories
Ry=1.1;
NumB=3;     %Number of bays without leaning column
%PROPERTIES OF THE PLASTIC HINGES:
%type=2 GRAVITY ANALYSIS!!
%type=1 REDUCTION CAPACITY OF THE COLUMN !!

type=2;

%Member sizes:

%Input the size of the member for each story from bottom to top

switch NumS
    case 2

    %TWO STORY BUILDING:

    beam_sizes = {'W30X132', 'W16X31'}';
    extcol_sizes = {'W24X131', 'W24X131'}';
    incol_sizes = {'W24X162', 'W24X162'}';

    ext_dblplate = [0.0, 0.0]';
    int_dblplate = [0.0, 0.0]';

    case 4

    %FOUR STORY BUILDING:

    beam_sizes = {'W21X73', 'W21X73', 'W21X57', 'W21X57'}';
    extcol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';
    incol_sizes = {'W24X103', 'W24X103', 'W24X103', 'W24X62'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0]';
    int_dblplate = [5/16, 5/16, 5/16, 5/16]';

    case 8
    %EIGHT STORY BUILDING:
```

```
    beam_sizes = {'W30X108', 'W30X116', 'W30X116', 'W27X94',
'W27X94','W27X94','W24X84','W21X68'}';
    extcol_sizes = {'W24X131', 'W24X131', 'W24X131',
'W24X131','W24X131','W24X131','W24X131','W24X94'}';
    incol_sizes = {'W24X162', 'W24X162', 'W24X162',
'W24X162','W24X162','W24X131','W24X131','W24X94'}';


    ext_dblplate = [1/16, 1/16, 1/16, 0.0, 0.0,  0.0,  0.0,  0.0]';
    int_dblplate = [9/16, 3/8, 11/16, 3/8, 9/16, 7/16, 9/16, 5/16]';

    case 12
    %TWELVE STORY BUILDING:

    beam_sizes = {'W30X124', 'W30X132', 'W30X132', 'W30X132',
'W30X116','W30X116','W30X116','W30X116','W27X94','W27X94','W24X84','W24X84'
}';
    extcol_sizes = {'W24X207', 'W24X207', 'W24X207', 'W24X162', 'W24X162',
'W24X146', 'W24X146', 'W24X131', 'W24X131', 'W24X131', 'W24X131',
'W24X84'}';
    incol_sizes =  {'W24X207', 'W24X207', 'W24X207', 'W24X207', 'W24X207',
'W24X176', 'W24X176', 'W24X162', 'W24X162', 'W24X131', 'W24X131',
'W24X94'}';

    ext_dblplate = [0.0, 0.0, 1/16, 1/16, 0.0, 0.0, 1/16, 1/16, 0.0, 0.0,
1/16, 1/16]';
    int_dblplate = [1/2, 7/16, 5/8, 5/8, 5/8, 5/8, 11/16, 11/16, 9/16,
9/16, 9/16, 9/16]';

    case 20
    %TWENTY STORY BUILDING:

    beam_sizes = {'W33X169', 'W33X169', 'W33X169', 'W33X169', 'W33X169',
'W33X169', 'W33X169', 'W33X169', 'W33X141', 'W33X141', 'W33X141',
'W33X141', 'W33X141', 'W33X141', 'W30X108', 'W30X108', 'W30X108',
'W30X108', 'W24X62', 'W24X62'}';
    extcol_sizes = {'W14X426', 'W14X426', 'W14X426', 'W14X426', 'W14X426',
'W14X398', 'W14X398', 'W14X370', 'W14X370', 'W14X311', 'W14X311',
'W14X283', 'W14X283', 'W14X233', 'W14X233', 'W14X159', 'W14X159',
'W14X132', 'W14X132', 'W14X132'}';
    incol_sizes =  {'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X335',
'W24X335', 'W24X335', 'W24X335', 'W24X335', 'W24X279', 'W24X279',
'W24X250', 'W24X250', 'W24X250', 'W24X250', 'W24X162', 'W24X162',
'W24X162', 'W24X162', 'W24X103'}';

    ext_dblplate = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1/16, 1/16, 1/4, 1/4, 3/8, 3/8, 0.0, 0.0]';
    int_dblplate = [1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 5/16, 5/16,
1/2, 1/2, 1/2, 1/2, 9/16, 9/16, 9/16, 9/16, 3/16, 3/16 ]';

end

%Material properties:

Fy = 55; % Nominal yield point (ksi)
E_steel = 29000; % ksi
nu_steel = 0.3;
```

```
G_steel = E_steel/(2*(1+nu_steel)); % ksi
%alpha = 0.03; % hardening in third segment of trilinear panel zone
material

%Read AISC.text file which contain mechanical properties of the cross
sections

AISC_ID = fopen('AISC.txt');
AISC_data = textscan(AISC_ID, '%s %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f');

beam_A      = zeros(NumS,1); % area
beam_d      = zeros(NumS,1); % depth
beam_bf     = zeros(NumS,1); % flange length
beam_tw     = zeros(NumS,1); % web thickness
beam_tf     = zeros(NumS,1); % flange thickness
beam_bf_2tf = zeros(NumS,1); % bf/2tf
beam_h_tw   = zeros(NumS,1); % h/tw
beam_Ix     = zeros(NumS,1);
beam_Zx     = zeros(NumS,1);
beam_Sx     = zeros(NumS,1);
beam_rx     = zeros(NumS,1);
beam_Iy     = zeros(NumS,1);
beam_Zy     = zeros(NumS,1);
beam_Sy     = zeros(NumS,1);
beam_ry     = zeros(NumS,1);
beam_J      = zeros(NumS,1);
beam_Aweb   = zeros(NumS,1); % equivalent to (d-2tf)*tw

extcol_A      = zeros(NumS,1);
extcol_d      = zeros(NumS,1);
extcol_bf     = zeros(NumS,1);
extcol_tw     = zeros(NumS,1);
extcol_tf     = zeros(NumS,1);
extcol_bf_2tf = zeros(NumS,1);
extcol_h_tw   = zeros(NumS,1);
extcol_Ix     = zeros(NumS,1);
extcol_Zx     = zeros(NumS,1);
extcol_Sx     = zeros(NumS,1);
extcol_rx     = zeros(NumS,1);
extcol_Iy     = zeros(NumS,1);
extcol_Zy     = zeros(NumS,1);
extcol_Sy     = zeros(NumS,1);
extcol_ry     = zeros(NumS,1);
extcol_J      = zeros(NumS,1);
extcol_Aweb   = zeros(NumS,1);

incol_A      = zeros(NumS,1);
incol_d      = zeros(NumS,1);
incol_bf     = zeros(NumS,1);
incol_tw     = zeros(NumS,1);
incol_tf     = zeros(NumS,1);
incol_bf_2tf = zeros(NumS,1);
incol_h_tw   = zeros(NumS,1);
incol_Ix     = zeros(NumS,1);
incol_Zx     = zeros(NumS,1);
incol_Sx     = zeros(NumS,1);
incol_rx     = zeros(NumS,1);
```

```
incol_Iy     = zeros(NumS,1);
incol_Zy     = zeros(NumS,1);
incol_Sy     = zeros(NumS,1);
incol_ry     = zeros(NumS,1);
incol_J      = zeros(NumS,1);
incol_Aweb   = zeros(NumS,1);

%Load cross sections properties of each story

for i = 1:NumS

beam_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);
beam_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},beam_sizes{i,1}),1);

end

for i = 1:NumS

extcol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
```

```
extcol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);
extcol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},extcol_sizes{i,1}),1);

incol_A(i,1)      =
AISC_data{1,3}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_d(i,1)      =
AISC_data{1,4}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf(i,1)     =
AISC_data{1,5}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tw(i,1)     =
AISC_data{1,6}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_tf(i,1)     =
AISC_data{1,7}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_bf_2tf(i,1) =
AISC_data{1,8}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_h_tw(i,1)   =
AISC_data{1,9}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Ix(i,1)     =
AISC_data{1,10}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zx(i,1)     =
AISC_data{1,11}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sx(i,1)     =
AISC_data{1,12}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_rx(i,1)     =
AISC_data{1,13}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Iy(i,1)     =
AISC_data{1,14}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Zy(i,1)     =
AISC_data{1,15}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_Sy(i,1)     =
AISC_data{1,16}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_ry(i,1)     =
AISC_data{1,17}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
incol_J(i,1)      =
AISC_data{1,18}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
```

```
incol_Aweb(i,1)   =
AISC_data{1,19}(strcmp(AISC_data{1,1},incol_sizes{i,1}),1);
end

%RBS settings

a1=0.625*beam_bf;
b1=0.75*beam_d;
c1=0.25*beam_bf;

a=0.625*beam_bf;
b=0.75*beam_d;
c=0.25*beam_bf;
%
%Plastic Moment at RBS location

RBS_Zx=beam_Zx-2*c.*beam_tf.*(beam_d-beam_tf);

%Max. Probable Moment at RBS

My_RBS=Ry*RBS_Zx*Fy;

sh_ext=a+b/2+extcol_d/2;
sh_int=a+b/2+incol_d/2;

%Beam Lengths:

beam_length_ext=bay-sh_ext-sh_int;
beam_length_int=bay-2*sh_int;


%Column Lengths:

h_first=h1-beam_d(1,1);

htypical=ones(NumS-1,1)*hi;
h_i(1)=h_first;


for i=2:NumS

 % h_i(i)=htypical(i-1)-beam_d(i-1,1)/2-beam_d(i,1)/2;
 h_i(i)=htypical(i-1)-beam_d(i-1);

end


nf=10;   %%This value of 10 comes from Ibarra's Dissertation. Pag 299.
I=ones(1,NumS);
Mp_My = 0.1;
Mc_My=1.1;
Lb=150;


if type==1
```

```
    [factor_ext,factor_int]=reduction_capacity(NumS,extcol_A,incol_A,Fy);
    reduction_capacity_ext=factor_ext;
    reduction_capacity_int=factor_int;
else

    reduction_capacity_ext=ones(1,NumS);
    reduction_capacity_int=ones(1,NumS);
end

for i=1:NumS

   Vy_ext(i)=2*(Ry*Fy*RBS_Zx(i)*1.15)/beam_length_ext(i)+(3.0274e-
01)*beam_length_ext(i)/2;

end

Ptotal=sum(Vy_ext);

for i=1:NumS

   Vy_int(i)=2*(Ry*Fy*incol_Zx(i))/beam_length_int(i)+2.9481e-
01*beam_length_int(i)/2;

end

for i=1:NumS


    %Spring properties of the columns (Non-RBS connections):
    %Exterior columns:

    ext_col_theta_p(i) =  0.087*(extcol_h_tw(i)^(-
0.365))*(extcol_bf_2tf(i)^(-
0.14))*((Lb/extcol_d(i))^(0.34))*(extcol_d(i)/21)^(-0.721)*(Fy/50)^(-0.23);
    ext_col_theta_pc(i) = 5.7*extcol_h_tw(i)^(-0.565)*extcol_bf_2tf(i)^(-
0.80)*(extcol_d(i)/21)^(-0.28)*(Fy/50)^(-0.43);
    ext_col_lambda(i) =   500*extcol_h_tw(i)^(-1.34)*extcol_bf_2tf(i)^(-
0.595)*(Fy/50)^(-0.36);

    Ke_ext_col(i) = (nf+1)*6.0*E_steel*extcol_Ix(i)/h_i(i);
    My_ext_col(i) = Ry*Fy*extcol_Zx(i);
    Kp_ext(i)=(Mc_My*My_ext_col(i)-My_ext_col(i))/(ext_col_theta_p(i));
    ass_ext_col(i) =Kp_ext(i)/Ke_ext_col(i);

    %Interior columns:

    int_col_theta_p(i) =  0.087*(incol_h_tw(i)^(-
0.365))*(incol_bf_2tf(i)^(-
0.14))*(Lb/(incol_d(i)))^(0.34)*(incol_d(i)/21)^(-0.721)*(Fy/50)^(-0.23);
    int_col_theta_pc(i) = 5.7*(incol_h_tw(i)^(-0.565))*(incol_bf_2tf(i)^(-
0.80))*(incol_d(i)/21)^(-0.28)*(Fy/50)^(-0.43);
    int_col_lambda(i) =   500*(incol_h_tw(i)^(-1.34))*(incol_bf_2tf(i)^(-
0.595))*(Fy/50)^(-0.36);

    Ke_int_col(i) = (nf+1)*6.0*E_steel*incol_Ix(i)/h_i(i);
    My_int_col(i) = Ry*Fy*incol_Zx(i);  %*red_fac;
```

```
    Kp_int(i)=(Mc_My*My_int_col(i)-My_int_col(i))/(int_col_theta_p(i));
    ass_int_col(i) =Kp_int(i)/Ke_int_col(i);


    % Spring properties of the beams (RBS connections):

    beam_theta_p(i) =  0.19*(beam_h_tw(i)^(-0.314))*(beam_bf_2tf(i)^(-
0.10))*(50)^(-0.1185) *(Lb/beam_d(i))^(0.113)*(beam_d(i)/21)^(-
0.76)*(Fy/50)^(-0.07);
    beam_theta_pc(i) = 9.62*beam_h_tw(i)^(-0.513)*beam_bf_2tf(i)^(-
0.863)*(50)^(-0.108) *(Fy/50)^(-0.36);
    beam_lambda(i) =   592*beam_h_tw(i)^(-1.138) *beam_bf_2tf(i)^(-
0.632)*(50)^(-0.205) *(Fy/50)^(-0.391);

    Ke_beam(i) = (nf+1)*6.0*E_steel*beam_Ix(i)*0.9/beam_length_int(i);
    Kp(i)=(Mc_My*My_RBS(i)-My_RBS(i))/(beam_theta_p(i));
    ass_beam(i) =Kp(i)/Ke_beam(i);


end

Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,sh_int,sh_ext);
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes,nf);
Nodesg(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,sh_int,sh_ext);
Sections(beam_sizes,NumS,extcol_sizes,incol_sizes,beam_d,beam_bf,beam_tf,be
am_tw,extcol_d,extcol_bf,extcol_tf,extcol_tw,incol_d,incol_bf,incol_tf,inco
l_tw)
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,inc
ol_bf,incol_tf,beam_d,ext_dblplate,int_dblplate,Ke_beam,ass_beam,My_RBS,bea
m_lambda,beam_theta_p,beam_theta_pc)
ZeroLength(NumS)
constraints(NumS)
masses(NumS)




%SPLICES INFO

%splices coord
%number of splices and long measured from the bottom of the column in that
%story
L1=63;
L2=64.55;
L3=64.55;
%Lx=--;

%Story Splice
Sp1=3;
Sp2=5;
Sp3=7;
```

```
%Spx=--;

%If there are more than 3 splices, add de respective lines (Lx and
Spx).Then add lines in "Draw the lines of the columns" section
%If there are less,put 0 in the respective "Sp" example: (Sp2=0).

%NODES

matrix1=importdata('Nodesg.tcl',' ',0);
A=matrix1.data;
nodes=A(:,2:3);


for j=1:NumS
for i=1:size(nodes,1)
    if nodes(i,2)==h1+hi*(i-1)-beam_d(j)
       nodes(i,1)=nodes(i,1);
    else if nodes(i,2)==h1+hi*(i-1)+beam_d(j)
       nodes(i,1)=nodes(i,1);
        else if nodes(i,2)==h1+hi*(i-1)
       nodes(i,1)=nodes(i,1);
    end
        end
    end
end
end



% Plot all the nodes:
figure('Name','Structure','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on


%COLUMNS

%leaning column
for j=1:NumS
    if j==1
yc2Lc(j)=h1;
yc1Lc(j)=0;

    else
        yc2Lc(j)=h1+hi*(j-1);
        yc1Lc(j)=h1+hi*(j-2);

    end
end

%Normal columns

for j=1:NumS
    if j==1
yc2(j)=h1-beam_d(j)/2;
yc1(j)=0;
```

```
    else
        yc2(j)=h1+hi*(j-1)-beam_d(j)/2;
        yc1(j)=h1+hi*(j-2)+beam_d(j-1)/2;

    end
end
for j=1:NumS+1

    if j==1
        for i=1:3
        yf2(i)=(h1/36)*(i);

        end
        yf22{j}=yf2;
    else if j==NumS+1
            for i=1:3
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
        yf11{j}=yf1;
    else if j==2
            for i=1:3
        yf2(i)=h1+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1-(beam_d(j-1)/2)-(h1/36)*(i);
            end
            yf22{j}=yf2;
            yf11{j}=yf1;
        else
            for i=1:3
        yf2(i)=h1+hi*(j-2)+(beam_d(j-1)/2)+(hi/36)*(i);
        yf1(i)=h1+hi*(j-2)-(beam_d(j-1)/2)-(hi/36)*(i);
            end
             yf22{j}=yf2;
             yf11{j}=yf1;
        end
        end
    end


end

a=0;
xc1=0;
xc2=0;


%Draw the lines of the columns

aa=0;
for j=1:(NumB+2)    %Number of columns including leaning column
    if j==(NumB+2)
        line([xc1+aa,xc2+aa], [yc1Lc(1),
yc2Lc(1)],'color','black','LineWidth',2.9)
    else
```

```
            line([xc1+aa,xc2+aa], [yc1(1),
yf22{1,1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,1),
yf22{1,1}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf22{1,1}(1,2),
yf22{1,1}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+aa,xc2+aa], [yc2(1),
yf11{1,2}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+aa,xc2+aa], [yf11{1,2}(1,1),
yf11{1,2}(1,2)],'color','black','LineWidth',2.9)
             line([xc1+aa,xc2+aa], [yf11{1,2}(1,2),
yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+aa,xc2+aa],
[yf22{1,1}(1,3),yf11{1,2}(1,3)],'color','black','LineWidth',2.9)

    end
    aa=aa+bay;
end

for j=1:(NumB+2)
    if j==(NumB+2)
        for i=2:NumS
            line([xc1+a,xc2+a], [yc1Lc(i),
yc2Lc(i)],'color','black','LineWidth',2.9)
        end
    else

    for i=2:NumS

        if i==Sp1

            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L1],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L1,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

        else if i==Sp2
            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
```

```
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L2],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L2,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)




            else if i==Sp3

           line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),yf11{1,i+1}(1,1)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),yf11{1,i+1}(1,2)
],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),yf11{1,i+1}(1,3)
],'color','black','LineWidth',2.9)


            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yc1(i)+L3],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc1(i)+L3,
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            % add the same syntanx than before if there are more than 3
            % splices.

            % EXAMPLE:

%             else if i==Spx
%             line([xc1+a,xc2+a], [yc1(i),
yc1(i)+Lx],'color','black','LineWidth',2.9)
%             line([xc1+a,xc2+a], [yc1(i)+Lx,
yc2(i)],'color','black','LineWidth',2.9)



               else
```

```
            line([xc1+a,xc2+a], [yc1(i),
yf22{1,i}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,1),
yf22{1,i}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf22{1,i}(1,2),
yf22{1,i}(1,3)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yc2(i),
yf11{1,i+1}(1,1)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,1),
yf11{1,i+1}(1,2)],'color','black','LineWidth',2.9)
            line([xc1+a,xc2+a], [yf11{1,i+1}(1,2),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)

            line([xc1+a,xc2+a], [yf22{1,i}(1,3),
yf11{1,i+1}(1,3)],'color','black','LineWidth',2.9)


                end
            end
        end

        %close the same "if" lines added

        %EXAMPLE
        %end


    end
    end


    a=a+bay;
end

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end




%BEAMS

xob1=0;
```

```
xob2=bay;


for j=1:NumS
    for i=1:(NumB+1) % 4 bays considering leaning column
        if j==pos_splices(j)
             if i==1 %Left exterior bay
    xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;




    xb2(i)=i*xob2-incol_d(j+1)/2;






    else
        if i==NumB  %Right exterior bay
            xb1(i)=(i-1)*xob2+incol_d(j+1)/2;




            xb2(i)=i*xob2-extcol_d(j+1)/2;




        else if i==(NumB+1)
           xb1(i)=(i-1)*xob2+extcol_d(j+1)/2;
            xb2(i)=i*xob2;
            else
                xb1(i)=(i-1)*xob2+incol_d(j+1)/2;




                xb2(i)=i*xob2-incol_d(j+1)/2;



            end
```

```
        end
    end
```

```
else
```

```
    if i==1 %Left exterior bay
    xb1(i)=(i-1)*xob2+extcol_d(j)/2;
```

```
    xb2(i)=i*xob2-incol_d(j)/2;
```

```
    else
        if i==NumB  %Right exterior bay
            xb1(i)=(i-1)*xob2+incol_d(j)/2;
```

```
            xb2(i)=i*xob2-extcol_d(j)/2;
```

```
        else if i==(NumB+1)
           xb1(i)=(i-1)*xob2+extcol_d(j)/2;
            xb2(i)=i*xob2;
            else
                xb1(i)=(i-1)*xob2+incol_d(j)/2;
```

```
                xb2(i)=i*xob2-incol_d(j)/2;



            end
        end
    end
end
    xpi{j}=xb1;
    xpf{j}=xb2;


end
end




for i=1:NumB
for k=1:7

    if i==1

    xbf4{k}=(i-1)*xob2+(extcol_d./2)+a1+(k-1)*b1./6;

    xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;

    else if i==NumB
    xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

     xbf3{k}=i*xob2-(extcol_d./2)-a1-(k-1)*b1./6;

        else
            xbf4{k}=(i-1)*xob2+(incol_d./2)+a1+(k-1)*b1./6;

            xbf3{k}=i*xob2-(incol_d./2)-a1-(k-1)*b1./6;



end



    end



xbf44{i}=xbf4;
xbf33{i}=xbf3;
```

```
end
end




yb1(1)=h1;
yb2(1)=h1;

for i=2:NumS
    yb1(i)=yb1(1)+hi*(i-1);
    yb2(i)=yb1(i);
end



% Draw the lines of the beams
redext=zeros(NumS,1);
redin=zeros(NumS,1);

for i=1:NumS

    if i==pos_splices(i)

        redext(i,1)=abs(extcol_d(i+1,1)/2-extcol_d(i,1)/2);
        redin(i,1)=abs(incol_d(i+1,1)/2-incol_d(i,1)/2);


end
end
for k=1:7
xbf44n{1,1}{1,k}=xbf44{1,1}{1,k}-redext;
xbf44n{1,2}{1,k}=xbf44{1,2}{1,k}-redin;
xbf44n{1,3}{1,k}=xbf44{1,3}{1,k}-redin;
xbf33n{1,1}{1,k}=xbf33{1,1}{1,k}+redin;
xbf33n{1,2}{1,k}=xbf33{1,2}{1,k}+redin;
xbf33n{1,3}{1,k}=xbf33{1,3}{1,k}+redext;
end

for i=1:NumS
    for j=1:NumB


        line([xpi{1,i}(1,j),xbf44n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,1}(i,1),xbf44n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,2}(i,1),xbf44n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
```

```
        line([xbf44n{1,j}{1,3}(i,1),xbf44n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,4}(i,1),xbf44n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,5}(i,1),xbf44n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf44n{1,j}{1,6}(i,1),xbf44n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)

        line([xpf{1,i}(1,j),xbf33n{1,j}{1,1}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,1}(i,1),xbf33n{1,j}{1,2}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,2}(i,1),xbf33n{1,j}{1,3}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,3}(i,1),xbf33n{1,j}{1,4}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,4}(i,1),xbf33n{1,j}{1,5}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,5}(i,1),xbf33n{1,j}{1,6}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)
        line([xbf33n{1,j}{1,6}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        line([xbf44n{1,j}{1,7}(i,1),xbf33n{1,j}{1,7}(i,1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)


        end









    mm1=NumB+1;

     line([xpi{1,i}(1,mm1),xpf{1,i}(1,mm1)], [yb1(i),
yb2(i)],'color','black','LineWidth',2.9)



end
```

```
%PANEL ZONES

for j=1:NumS

    if j==pos_splices(j)

          for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j+1)/2;
x2p(i)=bay*(i-1)+extcol_d(j+1)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j+1)/2;
            x2p(i)=bay*(i-1)+incol_d(j+1)/2;
        end
    end


    else



    for i=1:(NumB+1)
        if i==1 || i==(NumB+1)
x1p(i)=bay*(i-1)-extcol_d(j)/2;
x2p(i)=bay*(i-1)+extcol_d(j)/2;

        else
            x1p(i)=bay*(i-1)-incol_d(j)/2;
            x2p(i)=bay*(i-1)+incol_d(j)/2;
        end
    end
    end
        x1pp{j}=x1p;
        x2pp{j}=x2p;
end
for i=1:(NumB+1)

x3p(i)=bay*(i-1);
end

for j=1:NumS

    yp1(j)=h1+hi*(j-1)-beam_d(j)/2;
    yp2(j)=h1+hi*(j-1)+beam_d(j)/2;
    yp3(j)=h1+hi*(j-1);
end



% Draw the lines of the panel zones

for i=1:NumS
    for j=1:(NumB+1)
```

```
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x1pp{1,i}(1,j), x1pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);

        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp3(i)],'color','black','LineWidth',2.9);
        line([x2pp{1,i}(1,j), x2pp{1,i}(1,j)], [yp3(i),
yp2(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp1(i),
yp1(i)],'color','black','LineWidth',2.9);

        line([x1pp{1,i}(1,j), x3p(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);
        line([x3p(1,j), x2pp{1,i}(1,j)], [yp2(i),
yp2(i)],'color','black','LineWidth',2.9);

    end
%     k=k-1;
end

%Plot all the nodes:
figure('Name','Nodes','NumberTitle','off','Color','white')
plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color','black')
hold on
```

ANEXO B1: FUNCIÓN MATLAB DE ELEMENTOS PARA MODELOS L/12 FIBER HINGE

```
function
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

%n=nf;

% bs=beam_sizes;
%
[BST0, BST1, BST2]=BeamSecTag(beam_sizes,NumS);
% [BSTT0, BSTT1, BSTT2]=BeamSecTag2(beam_sizes,NumS);
[CSTEXT,CSTIN]=ColSecTag(extcol_sizes,incol_sizes,NumS);

fileID = fopen(['Elements','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# ELEMENTS OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');



fprintf(fileID, '%s\r\n', '# set up geometric transformations of element');
% fprintf(fileID, '%s\r\n', '# separate columns and beams, in case of P-
Delta analysis for columns');
% fprintf(fileID, '%s\r\n', '# all columns');
% fprintf(fileID, '%s\r\n', 'set transfTag 1');
% fprintf(fileID, '%s\r\n', '# all beams');
fprintf(fileID, '%s\r\n', 'set IDBeamTransf 2');
fprintf(fileID, '%s\r\n', 'set IDColTransf 3');
% fprintf(fileID, '%s\r\n', '# options, Linear PDelta Corotational ');
fprintf(fileID, '%s\r\n', 'set ColTransfType PDelta ');
% fprintf(fileID, '%s\r\n', '# only columns can have PDelta effects
(gravity effects)');
fprintf(fileID, '%s\r\n', 'geomTransf $ColTransfType $IDColTransf');
fprintf(fileID, '%s\r\n', 'geomTransf PDelta $IDBeamTransf');
fprintf(fileID, '%s\r\n', '# number of Integration points');
fprintf(fileID, '%s\r\n', 'set npC 3');
fprintf(fileID, '%s\r\n', 'set npB 3');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', '# Integration: Lobatto, Radau, Legendre,
NewtonCotes');

fprintf(fileID, '%s\r\n', 'set IntegrationC Lobatto');
fprintf(fileID, '%s\r\n', 'set IntegrationB Lobatto');

% fprintf(fileID, '%s\r\n', '# Beam Section Tag');
%
% for i=1:NumS
%
```

```
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT0{i,1}, num2str(i+10));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT1{i,1}, num2str(i+20));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT2{i,1}, num2str(i+30));
%
%
%
% end
%
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '# Column Section Tag');
%
% for i=1:NumS
%
%
% fprintf(fileID, '%s %17s %5s \r\n', 'set',
strcat(extcol_sizes{i,1},'aa',num2str(i),'aa','ext'), num2str(i+100));
% fprintf(fileID, '%s %17s %5s\r\n', 'set',
strcat(incol_sizes{i,1},'aa',num2str(i),'aa','in '), num2str(i+200));
% end


%COLUMN ELEMENTS:

a=3;
z=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;
    z=z+1;
end



for i=1:NumS

     if i==pos_splices(i)
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN ELASTIC ELEMENTS (WITH
SPLICES) FOR THE STORY: ', i);
        fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn
$eleID $iNode $jNode $A $E $I $transfID');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xya" where 1 =
col, x = Pier #, y = Story #, a= 91,92');
        fprintf(fileID, '%s\r\n', '');

         for j=1:NumB+1
            for l=91:92
                if j==1 || j==NumB+1
                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
```

```
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i), num2str(l)]);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
extcol_A(i), 29000, extcol_Ix(i), '$transfTag');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
extcol_A(i+1), 29000, extcol_Ix(i+1), '$transfTag');

                    end
                else

                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i),num2str(l)]);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
incol_A(i), 29000, incol_Ix(i), '$transfTag');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1),'0', '3', '1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
incol_A(i+1), 29000, incol_Ix(i+1), '$transfTag');


                    end



                end

            end

        end

    else
        fprintf(fileID, '%s\r\n', '');
       fprintf(fileID, '%s %u\r\n', '#COLUMN ELASTIC ELEMENTS FOR THE
STORY: ', i);
```

```
        fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn
$eleID $iNode $jNode $A $E $I $transfID');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xy" where 1 = col,
x = Pier #, y = Story #');
        fprintf(fileID, '%s\r\n', '');

        for j=1:NumB+1
         if j==1 || j==NumB+1

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3','1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
extcol_A(i), 29000, extcol_Ix(i), '$transfTag');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
incol_A(i), 29000, incol_Ix(i), '$transfTag');
         end
        end
     end

end

for i=1:NumS
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN FIBER ELEMENTS FOR THE STORY:
', i);
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xyz" where 1 =
col, x = Pier #, y = Story #, z=position of the fiber (1-3), 2= below
1=above');
        fprintf(fileID, '%s\r\n', '');

  if i== pos_splices(i)

        for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
```

```
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1,
node2,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2,
node3,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3,
node4,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



            end
        end

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
```

```
            else

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');

            end
          end

        else

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
```

```
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



            end
           end

         for j=1:NumB+1
%       end
%
% end


for i=2:NumS+1

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#BEAM ELASTIC ELEMENTS FOR THE FLOOR: ',
i);
     fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn $eleID
$iNode $jNode $A $E $I $transfID');
     fprintf(fileID, '%s\r\n', '#eleID convention:  "2xy" where 2 = beams,
x = Bay #, y = Floor #, 111 = left beam border, 222= right beam border,
333= center beam');
     fprintf(fileID, '%s\r\n', '');

     for j=1:NumB


            BeamTag1=(['2',num2str(j),num2str(i),'111']);%num2str =
converts a numeric array into a character array
            BeamTag2=(['2',num2str(j+1),num2str(i),'222']);
            BeamTag3=(['2',num2str(j+1),num2str(i),'333']);

            nodeT1_1=([num2str(j), num2str(i), '005']);
            nodeT1_2=([num2str(j), num2str(i), '0','1','4']);
            nodeT2_1=([num2str(j+1), num2str(i),'100']);
            nodeT2_2=([num2str(j+1), num2str(i),'0', '1', '3']);
            nodeT3_1=([num2str(j), num2str(i), '0','7','4']);
            nodeT3_2=([num2str(j+1), num2str(i), '0','7','3']);



            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag1, nodeT1_1, nodeT1_2,
beam_A(i-1),29000, beam_Ix(i-1), '$transfTag');
```

```
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag2, nodeT2_1, nodeT2_2,
beam_A(i-1),29000, beam_Ix(i-1), '$transfTag');

            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag3, nodeT3_1, nodeT3_2,
beam_A(i-1),29000, beam_Ix(i-1), '$transfTag');
     end

end




for i=2:NumS+1
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#BEAM RBS ELEMENTS FOR THE FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn $eleTag
$iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
     fprintf(fileID, '%s\r\n', '#eleID convention:  "2xyz" where 2 = beams,
x = Bay #, y = Floor # z=position of the rbs element from joint to beam
direction (1-7) 4= left 3= right');
     fprintf(fileID, '%s\r\n', '');

     for j=1:NumB


            BeamTag41=(['2',num2str(j),num2str(i),'1', '4']);
            BeamTag42=(['2',num2str(j),num2str(i),'2', '4']);
            BeamTag43=(['2',num2str(j),num2str(i),'3', '4']);
            BeamTag44=(['2',num2str(j),num2str(i),'4', '4']);
            BeamTag45=(['2',num2str(j),num2str(i),'5', '4']);
            BeamTag46=(['2',num2str(j),num2str(i),'6', '4']);



            nodeTag4_1=([num2str(j), num2str(i), '0','1','4']); %num2str =
converts a numeric array into a character array
            nodeTag4_2=([num2str(j), num2str(i), '0','2','4']);
             nodeTag4_3=([num2str(j), num2str(i), '0','3','4']);
              nodeTag4_4=([num2str(j), num2str(i), '0','4','4']);
               nodeTag4_5=([num2str(j), num2str(i), '0','5','4']);
                nodeTag4_6=([num2str(j), num2str(i), '0','6','4']);
                 nodeTag4_7=([num2str(j), num2str(i), '0','7','4']);



            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag41, nodeTag4_1,
nodeTag4_2,'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"' );
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag42, nodeTag4_2,
nodeTag4_3,'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag43, nodeTag4_3,
nodeTag4_4,'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
```

```
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag44, nodeTag4_4,
nodeTag4_5,'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag45, nodeTag4_5,
nodeTag4_6,'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag46, nodeTag4_6,
nodeTag4_7,'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"');


            BeamTag31=['2',num2str(j),num2str(i),'1', '3'];
            BeamTag32=['2',num2str(j),num2str(i),'2', '3'];
            BeamTag33=['2',num2str(j),num2str(i),'3', '3'];
            BeamTag34=['2',num2str(j),num2str(i),'4', '3'];
            BeamTag35=['2',num2str(j),num2str(i),'5', '3'];
            BeamTag36=['2',num2str(j),num2str(i),'6', '3'];


            nodeTag3_1=[num2str(j+1), num2str(i), '0','1','3']; %num2str
= converts a numeric array into a character array
            nodeTag3_2=[num2str(j+1), num2str(i), '0','2','3'];
            nodeTag3_3=[num2str(j+1), num2str(i), '0','3','3'];
            nodeTag3_4=[num2str(j+1), num2str(i), '0','4','3'];
            nodeTag3_5=[num2str(j+1), num2str(i), '0','5','3'];
            nodeTag3_6=[num2str(j+1), num2str(i), '0','6','3'];
            nodeTag3_7=[num2str(j+1), num2str(i), '0','7','3'];


            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag31, nodeTag3_1, nodeTag3_2,
'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag32, nodeTag3_2, nodeTag3_3,
'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag33, nodeTag3_3, nodeTag3_4,
'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag34, nodeTag3_4, nodeTag3_5,
'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag35, nodeTag3_5, nodeTag3_6,
'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag36, nodeTag3_6, nodeTag3_7,
'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');




    end
end
```

```
%TRUSS ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#TRUSS ELEMENTS FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '#command: element truss $eleID $iNode $jNode $A
$materialID');
fprintf(fileID, '%s\r\n', '#eleID convention: 6xy, 6 = truss link, x = Bay
#, y = Floor #');
fprintf(fileID, '%s\r\n', 'set TrussMatID  600');
fprintf(fileID, '%s\r\n', 'set Arigid  1000.0');
fprintf(fileID, '%s\r\n', 'set Irigid  0.0001');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic   $TrussMatID
29000');


for i=2:NumS+1

     fprintf(fileID, '%s %u\r\n', '','');
     TrussTag=(['6','4',num2str(i)]);
     node1 = (['4', num2str(i),'005']); %num2str = converts a numeric array
into a character array
     node2 = (['5', num2str(i)]);
     fprintf(fileID, '%13s %6s %6s %6s %21s %12s\r\n', 'element truss',
TrussTag, node1, node2, '  $Arigid   $TrussMatID');

end

%LINK ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#LEANING COLUMN FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '# eleID convention:  7xy, 7 = p-delta columns, x
= Pier #, y = Story #');
fprintf(fileID, '%s\r\n', '');

for i=1:NumS


 %fprintf(fileID, '%s %u\r\n', '','');
 LeaningTag=(['7','5',num2str(i)]);
 node1 = (['5',num2str(i)]); %num2str = converts a numeric array into a
character array
 node2 = (['5',num2str(i+1)]);
 fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e %12s\r\n', 'element
elasticBeamColumn', LeaningTag, node1, node2, 10000000,29000, 0.001,
'$transfTag');


end


%PANEL ZONE ELEMENTS:


for i=2:NumS+1
```

```
    for j=1:NumB+1


            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', '#ELEMENTS FOR THE PANEL ZONE
FLOOR:',i,' #PIER: ',j);
            fprintf(fileID, '%s\r\n', '# eleID convention:  500xya, 500 =
panel zone element, x = Pier #, y = Floor #');
            fprintf(fileID, '%s\r\n', '# "a" convention: defined in
elemPanelZone2D.tcl, but 1 = top left element');
            fprintf(fileID, '%s\r\n', '');

            PanelElemTag1=(['500',num2str(j),num2str(i),'1']);
            node1 = ([num2str(j),num2str(i),'002']); %num2str = converts a
numeric array into a character array
            node2 = ([num2str(j),num2str(i),'7']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag1, node1, node2,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag2=(['500',num2str(j),num2str(i),'2']);
            node3 = ([num2str(j),num2str(i),'7']); %num2str = converts a
numeric array into a character array
            node4 = ([num2str(j),num2str(i),'003']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag2, node3, node4,
10000000,29000, 1000000, '$transfTag');


            PanelElemTag3=(['500',num2str(j),num2str(i),'3']);
            node5 = ([num2str(j),num2str(i),'004']); %num2str = converts a
numeric array into a character array
            node6 = ([num2str(j),num2str(i),'005']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag3, node5, node6,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag4=(['500',num2str(j),num2str(i),'4']);
            node7 = ([num2str(j),num2str(i),'005']); %num2str = converts a
numeric array into a character array
            node8 = ([num2str(j),num2str(i),'006']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag4, node7, node8,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag5=(['500',num2str(j),num2str(i),'5']);
            node9 = ([num2str(j),num2str(i),'007']); %num2str = converts a
numeric array into a character array
            node10 = ([num2str(j),num2str(i),'5']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag5, node9, node10,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag6=(['500',num2str(j),num2str(i),'6']);
            node11 = ([num2str(j),num2str(i),'5']); %num2str = converts a
numeric array into a character array
```

```
            node12 = ([num2str(j),num2str(i),'008']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag6, node11, node12,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag7=(['500',num2str(j),num2str(i),'7']);
            node13 = ([num2str(j),num2str(i),'009']); %num2str = converts a
numeric array into a character array
            node14 = ([num2str(j),num2str(i),'100']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag7, node13, node14,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag8=(['500',num2str(j),num2str(i),'8']);
            node15 = ([num2str(j),num2str(i),'100']); %num2str = converts a
numeric array into a character array
            node16 = ([num2str(j),num2str(i),'001']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag8, node15, node16,
10000000,29000, 1000000, '$transfTag');


    end
end

fclose(fileID);

end
```

ANEXO B2: FUNCIÓN MATLAB DE ELEMENTOS PARA MODELOS L/12 FIBER

```
function
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

%n=nf;

% bs=beam_sizes;
%
[BST0, BST1, BST2, BST]=BeamSecTag(beam_sizes,NumS);
% [BSTT0, BSTT1, BSTT2]=BeamSecTag2(beam_sizes,NumS);
[CSTEXT,CSTIN]=ColSecTag(extcol_sizes,incol_sizes,NumS);

fileID = fopen(['Elements','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# ELEMENTS OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');



fprintf(fileID, '%s\r\n', '# set up geometric transformations of element');
% fprintf(fileID, '%s\r\n', '# separate columns and beams, in case of P-
Delta analysis for columns');
% fprintf(fileID, '%s\r\n', '# all columns');
% fprintf(fileID, '%s\r\n', 'set transfTag 1');
% fprintf(fileID, '%s\r\n', '# all beams');
fprintf(fileID, '%s\r\n', 'set IDBeamTransf 2');
fprintf(fileID, '%s\r\n', 'set IDColTransf 3');
% fprintf(fileID, '%s\r\n', '# options, Linear PDelta Corotational ');
fprintf(fileID, '%s\r\n', 'set ColTransfType PDelta ');
% fprintf(fileID, '%s\r\n', '# only columns can have PDelta effects
(gravity effects)');
fprintf(fileID, '%s\r\n', 'geomTransf $ColTransfType $IDColTransf');
fprintf(fileID, '%s\r\n', 'geomTransf PDelta $IDBeamTransf');
fprintf(fileID, '%s\r\n', '# number of Integration points');
fprintf(fileID, '%s\r\n', 'set npC 3');
fprintf(fileID, '%s\r\n', 'set npB 3');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', '# Integration: Lobatto, Radau, Legendre,
NewtonCotes');

fprintf(fileID, '%s\r\n', 'set IntegrationC Lobatto');
fprintf(fileID, '%s\r\n', 'set IntegrationB Lobatto');
% fprintf(fileID, '%s\r\n', '# Beam Section Tag');
%
% for i=1:NumS
%
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT0{i,1}, num2str(i+10));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT1{i,1}, num2str(i+20));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT2{i,1}, num2str(i+30));
```

```
%
%
%
% end
%
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '# Column Section Tag');
%
% for i=1:NumS
%
%
% fprintf(fileID, '%s %17s %5s \r\n', 'set',
strcat(extcol_sizes{i,1},'aa',num2str(i),'aa','ext'), num2str(i+100));
% fprintf(fileID, '%s %17s %5s\r\n', 'set',
strcat(incol_sizes{i,1},'aa',num2str(i),'aa','in '), num2str(i+200));
% end


%COLUMN ELEMENTS:

a=3;
z=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;
    z=z+1;
end



for i=1:NumS

     if i==pos_splices(i)
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN CENTER ELEMENTS (WITH
SPLICES) FOR THE STORY: ', i);
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xya" where 1 =
col, x = Pier #, y = Story #, a= 91,92');
        fprintf(fileID, '%s\r\n', '');

         for j=1:NumB+1
            for l=91:92
                if j==1 || j==NumB+1
                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i), num2str(l)]);
```

```
                        fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf', '"$IntegrationC', CSTEXT{i,1},'$npC"' );


                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf', '"$IntegrationC', CSTEXT{i+1,1},'$npC"' );


                    end
                else

                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i),num2str(l)]);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf', '"$IntegrationC', CSTIN{i,1},'$npC"' );


                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1),'0', '3', '1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf', '"$IntegrationC', CSTIN{i+1,1},'$npC"' );

                    end



                end

            end

        end

    else
        fprintf(fileID, '%s\r\n', '');
       fprintf(fileID, '%s %u\r\n', '#COLUMN CENTER ELEMENTS FOR THE
STORY: ', i);
```

```
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xy" where 1 = col,
x = Pier #, y = Story #');
        fprintf(fileID, '%s\r\n', '');

        for j=1:NumB+1
         if j==1 || j==NumB+1

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3','1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf', '"$IntegrationC', CSTEXT{i,1},'$npC"' );


         else

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf', '"$IntegrationC', CSTIN{i,1},'$npC"' );



         end
        end
     end

end

for i=1:NumS
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN FIBER ELEMENTS FOR THE STORY:
', i);
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xyz" where 1 =
col, x = Pier #, y = Story #, z=position of the fiber (1-3), 2= below
1=above');
        fprintf(fileID, '%s\r\n', '');

  if i== pos_splices(i)

        for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
```

```
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
               fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1,
node2,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2,
node3,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3,
node4,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



             end
        end

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                 fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
```

```
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');

            else

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');

            end
          end

        else

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
```

```
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



            end
           end

         for j=1:NumB+1

             if j==1 || j==NumB+1

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

             else

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
```

```
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');

            end



        end







    end
end



for i=2:NumS+1

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#BEAM FIBER AND ELASTIC ELEMENTS FOR THE
FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn $eleTag
$iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
     fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn $eleID
$iNode $jNode $A $E $I $transfID');
     fprintf(fileID, '%s\r\n', '#eleID convention:  "2xy" where 2 = beams,
x = Bay #, y = Floor #, 111 = left beam border, 222= right beam border,
333= center beam');
     fprintf(fileID, '%s\r\n', '');

     for j=1:NumB


            BeamTag1=(['2',num2str(j),num2str(i),'111']);%num2str =
converts a numeric array into a character array
            BeamTag2=(['2',num2str(j+1),num2str(i),'222']);
            BeamTag3=(['2',num2str(j+1),num2str(i),'333']);

            nodeT1_1=([num2str(j), num2str(i), '005']);
            nodeT1_2=([num2str(j), num2str(i), '0','1','4']);
            nodeT2_1=([num2str(j+1), num2str(i),'100']);
            nodeT2_2=([num2str(j+1), num2str(i),'0', '1', '3']);
            nodeT3_1=([num2str(j), num2str(i), '0','7','4']);
            nodeT3_2=([num2str(j+1), num2str(i), '0','7','3']);



            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag1, nodeT1_1, nodeT1_2,
beam_A(i-1),29000, beam_Ix(i-1), '$transfTag');
```

```
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag2, nodeT2_1, nodeT2_2,
beam_A(i-1),29000, beam_Ix(i-1), '$transfTag');

            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag3, nodeT3_1, nodeT3_2,'$IDBeamTransf',
'"$IntegrationB', BST{i-1,1},'$npB"' );

     end

end




for i=2:NumS+1
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#BEAM RBS ELEMENTS FOR THE FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn $eleTag
$iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
     fprintf(fileID, '%s\r\n', '#eleID convention:  "2xyz" where 2 = beams,
x = Bay #, y = Floor # z=position of the rbs element from joint to beam
direction (1-7) 4= left 3= right');
     fprintf(fileID, '%s\r\n', '');

     for j=1:NumB


            BeamTag41=(['2',num2str(j),num2str(i),'1', '4']);
            BeamTag42=(['2',num2str(j),num2str(i),'2', '4']);
            BeamTag43=(['2',num2str(j),num2str(i),'3', '4']);
            BeamTag44=(['2',num2str(j),num2str(i),'4', '4']);
            BeamTag45=(['2',num2str(j),num2str(i),'5', '4']);
            BeamTag46=(['2',num2str(j),num2str(i),'6', '4']);



            nodeTag4_1=([num2str(j), num2str(i), '0','1','4']); %num2str =
converts a numeric array into a character array
            nodeTag4_2=([num2str(j), num2str(i), '0','2','4']);
             nodeTag4_3=([num2str(j), num2str(i), '0','3','4']);
              nodeTag4_4=([num2str(j), num2str(i), '0','4','4']);
               nodeTag4_5=([num2str(j), num2str(i), '0','5','4']);
                nodeTag4_6=([num2str(j), num2str(i), '0','6','4']);
                 nodeTag4_7=([num2str(j), num2str(i), '0','7','4']);


            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag41, nodeTag4_1,
nodeTag4_2,'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"' );
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag42, nodeTag4_2,
nodeTag4_3,'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');
```

```
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag43, nodeTag4_3,
nodeTag4_4,'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag44, nodeTag4_4,
nodeTag4_5,'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag45, nodeTag4_5,
nodeTag4_6,'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag46, nodeTag4_6,
nodeTag4_7,'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"');


            BeamTag31=(['2',num2str(j),num2str(i),'1', '3']);
            BeamTag32=(['2',num2str(j),num2str(i),'2', '3']);
            BeamTag33=(['2',num2str(j),num2str(i),'3', '3']);
            BeamTag34=(['2',num2str(j),num2str(i),'4', '3']);
            BeamTag35=(['2',num2str(j),num2str(i),'5', '3']);
            BeamTag36=(['2',num2str(j),num2str(i),'6', '3']);


            nodeTag3_1=([num2str(j+1), num2str(i), '0','1','3']); %num2str
= converts a numeric array into a character array
            nodeTag3_2=([num2str(j+1), num2str(i), '0','2','3']);
            nodeTag3_3=([num2str(j+1), num2str(i), '0','3','3']);
            nodeTag3_4=([num2str(j+1), num2str(i), '0','4','3']);
            nodeTag3_5=([num2str(j+1), num2str(i), '0','5','3']);
            nodeTag3_6=([num2str(j+1), num2str(i), '0','6','3']);
            nodeTag3_7=([num2str(j+1), num2str(i), '0','7','3']);


            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag31, nodeTag3_1, nodeTag3_2,
'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag32, nodeTag3_2, nodeTag3_3,
'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag33, nodeTag3_3, nodeTag3_4,
'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag34, nodeTag3_4, nodeTag3_5,
'$IDBeamTransf', '"$IntegrationB', BST0{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag35, nodeTag3_5, nodeTag3_6,
'$IDBeamTransf', '"$IntegrationB', BST2{i-1,1},'$npB"');
            fprintf(fileID, '%27s %6s %6s %6s %12s %12s %12s %4s\r\n',
'element forceBeamColumn', BeamTag36, nodeTag3_6, nodeTag3_7,
'$IDBeamTransf', '"$IntegrationB', BST1{i-1,1},'$npB"');




    end
```

```
end



%TRUSS ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#TRUSS ELEMENTS FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '#command: element truss $eleID $iNode $jNode $A
$materialID');
fprintf(fileID, '%s\r\n', '#eleID convention: 6xy, 6 = truss link, x = Bay
#, y = Floor #');
fprintf(fileID, '%s\r\n', 'set TrussMatID  600');
fprintf(fileID, '%s\r\n', 'set Arigid  1000.0');
fprintf(fileID, '%s\r\n', 'set Irigid  0.0001');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic   $TrussMatID
29000');


for i=2:NumS+1

     fprintf(fileID, '%s %u\r\n', '','');
     TrussTag=(['6','4',num2str(i)]);
     node1 = (['4', num2str(i),'005']); %num2str = converts a numeric array
into a character array
     node2 = (['5', num2str(i)]);
     fprintf(fileID, '%13s %6s %6s %6s %21s %12s\r\n', 'element truss',
TrussTag, node1, node2, '  $Arigid   $TrussMatID');

end

%LINK ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#LEANING COLUMN FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '# eleID convention:  7xy, 7 = p-delta columns, x
= Pier #, y = Story #');
fprintf(fileID, '%s\r\n', '');

for i=1:NumS


 %fprintf(fileID, '%s %u\r\n', '','');
 LeaningTag=(['7','5',num2str(i)]);
 node1 = (['5',num2str(i)]); %num2str = converts a numeric array into a
character array
 node2 = (['5',num2str(i+1)]);
 fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e %12s\r\n', 'element
elasticBeamColumn', LeaningTag, node1, node2, 10000000,29000, 0.001,
'$transfTag');


end


%PANEL ZONE ELEMENTS:
```

```
for i=2:NumS+1

    for j=1:NumB+1


            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', '#ELEMENTS FOR THE PANEL ZONE
FLOOR:',i,' #PIER: ',j);
            fprintf(fileID, '%s\r\n', '# eleID convention:  500xya, 500 =
panel zone element, x = Pier #, y = Floor #');
            fprintf(fileID, '%s\r\n', '# "a" convention: defined in
elemPanelZone2D.tcl, but 1 = top left element');
            fprintf(fileID, '%s\r\n', '');

            PanelElemTag1=(['500',num2str(j),num2str(i),'1']);
            node1 = ([num2str(j),num2str(i),'002']); %num2str = converts a
numeric array into a character array
            node2 = ([num2str(j),num2str(i),'7']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag1, node1, node2,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag2=(['500',num2str(j),num2str(i),'2']);
            node3 = ([num2str(j),num2str(i),'7']); %num2str = converts a
numeric array into a character array
            node4 = ([num2str(j),num2str(i),'003']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag2, node3, node4,
10000000,29000, 1000000, '$transfTag');


            PanelElemTag3=(['500',num2str(j),num2str(i),'3']);
            node5 = ([num2str(j),num2str(i),'004']); %num2str = converts a
numeric array into a character array
            node6 = ([num2str(j),num2str(i),'005']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag3, node5, node6,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag4=(['500',num2str(j),num2str(i),'4']);
            node7 = ([num2str(j),num2str(i),'005']); %num2str = converts a
numeric array into a character array
            node8 = ([num2str(j),num2str(i),'006']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag4, node7, node8,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag5=(['500',num2str(j),num2str(i),'5']);
            node9 = ([num2str(j),num2str(i),'007']); %num2str = converts a
numeric array into a character array
            node10 = ([num2str(j),num2str(i),'5']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag5, node9, node10,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag6=(['500',num2str(j),num2str(i),'6']);
```

```
            node11 = ([num2str(j),num2str(i),'5']); %num2str = converts a
numeric array into a character array
            node12 = ([num2str(j),num2str(i),'008']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag6, node11, node12,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag7=(['500',num2str(j),num2str(i),'7']);
            node13 = ([num2str(j),num2str(i),'009']); %num2str = converts a
numeric array into a character array
            node14 = ([num2str(j),num2str(i),'100']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag7, node13, node14,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag8=(['500',num2str(j),num2str(i),'8']);
            node15 = ([num2str(j),num2str(i),'100']); %num2str = converts a
numeric array into a character array
            node16 = ([num2str(j),num2str(i),'001']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag8, node15, node16,
10000000,29000, 1000000, '$transfTag');


    end
end

fclose(fileID);

end
```

ANEXO B3: FUNCIÓN MATLAB DE ELEMENTOS PARA MODELOS L/12 FIBER HINGE AND CPH

```
function
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes,nf)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

n=nf;

% bs=beam_sizes;
%
[BST0, BST1, BST2]=BeamSecTag(beam_sizes,NumS);
% [BSTT0, BSTT1, BSTT2]=BeamSecTag2(beam_sizes,NumS);
[CSTEXT,CSTIN]=ColSecTag(extcol_sizes,incol_sizes,NumS);

fileID = fopen(['Elements','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# ELEMENTS OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');



fprintf(fileID, '%s\r\n', '# set up geometric transformations of element');
% fprintf(fileID, '%s\r\n', '# separate columns and beams, in case of P-
Delta analysis for columns');
% fprintf(fileID, '%s\r\n', '# all columns');
% fprintf(fileID, '%s\r\n', 'set transfTag 1');
% fprintf(fileID, '%s\r\n', '# all beams');
fprintf(fileID, '%s\r\n', 'set IDBeamTransf 2');
fprintf(fileID, '%s\r\n', 'set IDColTransf 3');
% fprintf(fileID, '%s\r\n', '# options, Linear PDelta Corotational ');
fprintf(fileID, '%s\r\n', 'set ColTransfType PDelta ');
% fprintf(fileID, '%s\r\n', '# only columns can have PDelta effects
(gravity effects)');
fprintf(fileID, '%s\r\n', 'geomTransf $ColTransfType $IDColTransf');
fprintf(fileID, '%s\r\n', 'geomTransf PDelta $IDBeamTransf');
fprintf(fileID, '%s\r\n', '# number of Integration points');
fprintf(fileID, '%s\r\n', 'set npC 3');
fprintf(fileID, '%s\r\n', 'set npB 3');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', '# Integration: Lobatto, Radau, Legendre,
NewtonCotes');

fprintf(fileID, '%s\r\n', 'set IntegrationC Lobatto');
fprintf(fileID, '%s\r\n', 'set IntegrationB Lobatto');

% fprintf(fileID, '%s\r\n', '# Beam Section Tag');
%
% for i=1:NumS
%
```

```
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT0{i,1}, num2str(i+10));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT1{i,1}, num2str(i+20));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT2{i,1}, num2str(i+30));
%
%
%
% end
%
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '# Column Section Tag');
%
% for i=1:NumS
%
%
% fprintf(fileID, '%s %17s %5s \r\n', 'set',
strcat(extcol_sizes{i,1},'aa',num2str(i),'aa','ext'), num2str(i+100));
% fprintf(fileID, '%s %17s %5s\r\n', 'set',
strcat(incol_sizes{i,1},'aa',num2str(i),'aa','in '), num2str(i+200));
% end


%COLUMN ELEMENTS:

a=3;
z=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;
    z=z+1;
end



for i=1:NumS

     if i==pos_splices(i)
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN ELASTIC ELEMENTS (WITH
SPLICES) FOR THE STORY: ', i);
        fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn
$eleID $iNode $jNode $A $E $I $transfID');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xya" where 1 =
col, x = Pier #, y = Story #, a= 91,92');
        fprintf(fileID, '%s\r\n', '');

         for j=1:NumB+1
            for l=91:92
                if j==1 || j==NumB+1
                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
```

```
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i), num2str(l)]);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
extcol_A(i), 29000, extcol_Ix(i), '$transfTag');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
extcol_A(i+1), 29000, extcol_Ix(i+1), '$transfTag');

                    end
                else

                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i),num2str(l)]);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
incol_A(i), 29000, incol_Ix(i), '$transfTag');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1),'0', '3', '1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
incol_A(i+1), 29000, incol_Ix(i+1), '$transfTag');


                    end



                end

            end

        end

    else
        fprintf(fileID, '%s\r\n', '');
       fprintf(fileID, '%s %u\r\n', '#COLUMN ELASTIC ELEMENTS FOR THE
STORY: ', i);
```

```
        fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn
$eleID $iNode $jNode $A $E $I $transfID');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xy" where 1 = col,
x = Pier #, y = Story #');
        fprintf(fileID, '%s\r\n', '');

        for j=1:NumB+1
         if j==1 || j==NumB+1

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3','1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
extcol_A(i), 29000, extcol_Ix(i), '$transfTag');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e
%12.4e %12s\r\n', 'element elasticBeamColumn', ColumnTag, node1, node2,
incol_A(i), 29000, incol_Ix(i), '$transfTag');
         end
        end
     end

end

for i=1:NumS
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN FIBER ELEMENTS FOR THE STORY:
', i);
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xyz" where 1 =
col, x = Pier #, y = Story #, z=position of the fiber (1-3), 2= below
1=above');
        fprintf(fileID, '%s\r\n', '');

  if i== pos_splices(i)

        for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
```

```
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1,
node2,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2,
node3,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3,
node4,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



            end
        end

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
```

```
            else

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');

            end
          end

        else

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
```

```
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



            end
           end

         for j=1:NumB+1

             if j==1 || j==NumB+1

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

             else

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
```

```
            end


        end




    end
end



for i=2:NumS+1

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#BEAM ELEMENTS FOR THE FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn $eleID
$iNode $jNode $A $E $I $transfID');
     fprintf(fileID, '%s\r\n', '#eleID convention:  "2xy" where 2 = beams,
x = Bay #, y = Floor #');
     fprintf(fileID, '%s\r\n', '');

     for j=1:NumB

            BeamRBSTag=(['2',num2str(j),num2str(i)]);
            BeamTag1=(['2',num2str(j),num2str(i),'11']);
            BeamTag2=(['2',num2str(j+1),num2str(i),'22']);
            nodeRBS1 = ([num2str(j), num2str(i),'2']); %num2str = converts
a numeric array into a character array
            nodeRBS2 = ([num2str(j+1), num2str(i),'3']);
            nodeT1_1=([num2str(j), num2str(i),'005']);
            nodeT1_2=([num2str(j), num2str(i),'1']);
            nodeT2_1=([num2str(j+1), num2str(i),'4']);
            nodeT2_2=([num2str(j+1), num2str(i),'100']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag1, nodeT1_1, nodeT1_2,
beam_A(i-1),29000, beam_Ix(i-1)*0.9, '$transfTag');
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamRBSTag, nodeRBS1, nodeRBS2,
beam_A(i-1),29000, beam_Ix(i-1)*(n+1)/n*0.9, '$transfTag');
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag2, nodeT2_1, nodeT2_2,
beam_A(i-1),29000, beam_Ix(i-1)*0.9, '$transfTag');
     end

end

%TRUSS ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#TRUSS ELEMENTS FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '#command: element truss $eleID $iNode $jNode $A
$materialID');
```

```
fprintf(fileID, '%s\r\n', '#eleID convention: 6xy, 6 = truss link, x = Bay
#, y = Floor #');
fprintf(fileID, '%s\r\n', 'set TrussMatID  600');
fprintf(fileID, '%s\r\n', 'set Arigid  1000.0');
fprintf(fileID, '%s\r\n', 'set Irigid  0.0001');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic   $TrussMatID
29000');


for i=2:NumS+1

     fprintf(fileID, '%s %u\r\n', '','');
     TrussTag=(['6','4',num2str(i)]);
     node1 = (['4', num2str(i),'005']); %num2str = converts a numeric array
into a character array
     node2 = (['5', num2str(i)]);
     fprintf(fileID, '%13s %6s %6s %6s %21s %12s\r\n', 'element truss',
TrussTag, node1, node2, '  $Arigid   $TrussMatID');

end

%LINK ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#LEANING COLUMN FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '# eleID convention:  7xy, 7 = p-delta columns, x
= Pier #, y = Story #');
fprintf(fileID, '%s\r\n', '');

for i=1:NumS


 %fprintf(fileID, '%s %u\r\n', '','');
 LeaningTag=(['7','5',num2str(i)]);
 node1 = (['5',num2str(i)]); %num2str = converts a numeric array into a
character array
 node2 = (['5',num2str(i+1)]);
 fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e %12s\r\n', 'element
elasticBeamColumn', LeaningTag, node1, node2, 10000000,29000, 0.001,
'$transfTag');


end


%PANEL ZONE ELEMENTS:


for i=2:NumS+1

    for j=1:NumB+1


            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', '#ELEMENTS FOR THE PANEL ZONE
FLOOR:',i,' #PIER: ',j);
```

```
            fprintf(fileID, '%s\r\n', '# eleID convention:  500xya, 500 =
panel zone element, x = Pier #, y = Floor #');
            fprintf(fileID, '%s\r\n', '# "a" convention: defined in
elemPanelZone2D.tcl, but 1 = top left element');
            fprintf(fileID, '%s\r\n', '');

            PanelElemTag1=(['500',num2str(j),num2str(i),'1']);
            node1 = ([num2str(j),num2str(i),'002']); %num2str = converts a
numeric array into a character array
            node2 = ([num2str(j),num2str(i),'7']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag1, node1, node2,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag2=(['500',num2str(j),num2str(i),'2']);
            node3 = ([num2str(j),num2str(i),'7']); %num2str = converts a
numeric array into a character array
            node4 = ([num2str(j),num2str(i),'003']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag2, node3, node4,
10000000,29000, 1000000, '$transfTag');


            PanelElemTag3=(['500',num2str(j),num2str(i),'3']);
            node5 = ([num2str(j),num2str(i),'004']); %num2str = converts a
numeric array into a character array
            node6 = ([num2str(j),num2str(i),'005']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag3, node5, node6,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag4=(['500',num2str(j),num2str(i),'4']);
            node7 = ([num2str(j),num2str(i),'005']); %num2str = converts a
numeric array into a character array
            node8 = ([num2str(j),num2str(i),'006']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag4, node7, node8,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag5=(['500',num2str(j),num2str(i),'5']);
            node9 = ([num2str(j),num2str(i),'007']); %num2str = converts a
numeric array into a character array
            node10 = ([num2str(j),num2str(i),'5']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag5, node9, node10,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag6=(['500',num2str(j),num2str(i),'6']);
            node11 = ([num2str(j),num2str(i),'5']); %num2str = converts a
numeric array into a character array
            node12 = ([num2str(j),num2str(i),'008']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag6, node11, node12,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag7=(['500',num2str(j),num2str(i),'7']);
```

```
            node13 = ([num2str(j),num2str(i),'009']); %num2str = converts a
numeric array into a character array
            node14 = ([num2str(j),num2str(i),'100']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag7, node13, node14,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag8=(['500',num2str(j),num2str(i),'8']);
            node15 = ([num2str(j),num2str(i),'100']); %num2str = converts a
numeric array into a character array
            node16 = ([num2str(j),num2str(i),'001']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag8, node15, node16,
10000000,29000, 1000000, '$transfTag');


    end
end

fclose(fileID);

end
```

ANEXO B4: FUNCIÓN MATLAB DE ELEMENTOS PARA MODELOS L/12 FIBER AND CPH

```
function
Elements(NumS,beam_A,beam_Ix,NumB,beam_sizes,extcol_A,extcol_Ix,incol_A,inc
ol_Ix,extcol_sizes,incol_sizes,nf)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

n=nf;

% bs=beam_sizes;
%
[BST0, BST1, BST2]=BeamSecTag(beam_sizes,NumS);
% [BSTT0, BSTT1, BSTT2]=BeamSecTag2(beam_sizes,NumS);
[CSTEXT,CSTIN]=ColSecTag(extcol_sizes,incol_sizes,NumS);

fileID = fopen(['Elements','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# ELEMENTS OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');



fprintf(fileID, '%s\r\n', '# set up geometric transformations of element');
% fprintf(fileID, '%s\r\n', '# separate columns and beams, in case of P-
Delta analysis for columns');
% fprintf(fileID, '%s\r\n', '# all columns');
% fprintf(fileID, '%s\r\n', 'set transfTag 1');
% fprintf(fileID, '%s\r\n', '# all beams');
fprintf(fileID, '%s\r\n', 'set IDBeamTransf 2');
fprintf(fileID, '%s\r\n', 'set IDColTransf 3');
% fprintf(fileID, '%s\r\n', '# options, Linear PDelta Corotational ');
fprintf(fileID, '%s\r\n', 'set ColTransfType PDelta ');
% fprintf(fileID, '%s\r\n', '# only columns can have PDelta effects
(gravity effects)');
fprintf(fileID, '%s\r\n', 'geomTransf $ColTransfType $IDColTransf');
fprintf(fileID, '%s\r\n', 'geomTransf PDelta $IDBeamTransf');
fprintf(fileID, '%s\r\n', '# number of Integration points');
fprintf(fileID, '%s\r\n', 'set npC 3');
fprintf(fileID, '%s\r\n', 'set npB 3');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', '# Integration: Lobatto, Radau, Legendre,
NewtonCotes');

fprintf(fileID, '%s\r\n', 'set IntegrationC Lobatto');
fprintf(fileID, '%s\r\n', 'set IntegrationB Lobatto');

% fprintf(fileID, '%s\r\n', '# Beam Section Tag');
%
% for i=1:NumS
%
```

```
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT0{i,1}, num2str(i+10));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT1{i,1}, num2str(i+20));
% fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT2{i,1}, num2str(i+30));
%
%
%
% end
%
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '# Column Section Tag');
%
% for i=1:NumS
%
%
% fprintf(fileID, '%s %17s %5s \r\n', 'set',
strcat(extcol_sizes{i,1},'aa',num2str(i),'aa','ext'), num2str(i+100));
% fprintf(fileID, '%s %17s %5s\r\n', 'set',
strcat(incol_sizes{i,1},'aa',num2str(i),'aa','in '), num2str(i+200));
% end


%COLUMN ELEMENTS:

a=3;
z=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;
    z=z+1;
end



for i=1:NumS

     if i==pos_splices(i)
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN CENTER ELEMENTS (WITH
SPLICES) FOR THE STORY: ', i);
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xya" where 1 =
col, x = Pier #, y = Story #, a= 91,92');
        fprintf(fileID, '%s\r\n', '');

         for j=1:NumB+1
            for l=91:92
                if j==1 || j==NumB+1
                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
```

```
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i), num2str(l)]);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');


                    end
                else

                    if l==91

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i),num2str(l)]);
                         fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                    else

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),num2str(l)]);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1),'0', '3', '1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');

                    end



                end

            end

        end

    else
        fprintf(fileID, '%s\r\n', '');
       fprintf(fileID, '%s %u\r\n', '#COLUMN CENTER ELEMENTS FOR THE
STORY: ', i);
```

```
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xy" where 1 = col,
x = Pier #, y = Story #');
        fprintf(fileID, '%s\r\n', '');

        for j=1:NumB+1
         if j==1 || j==NumB+1

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3','2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3','1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

                    else

                        ColumnTag=(['1',num2str(j),num2str(i)]);
                        node1 = ([num2str(j), num2str(i), '0', '3', '2']);
%num2str = converts a numeric array into a character array
                        node2 = ([num2str(j), num2str(i+1), '0', '3',
'1']);
                        fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s
%4s\r\n', 'element forceBeamColumn', ColumnTag, node1,
node2,'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
         end
        end
     end

end

for i=1:NumS
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN FIBER ELEMENTS FOR THE STORY:
', i);
        fprintf(fileID, '%s\r\n', '#command: element forceBeamColumn
$eleTag $iNode $jNode $transfTag "IntegrationType arg1 arg2 ..."');
        fprintf(fileID, '%s\r\n', '#eleID convention:  "1xyz" where 1 =
col, x = Pier #, y = Story #, z=position of the fiber (1-3), 2= below
1=above');
        fprintf(fileID, '%s\r\n', '');

  if i== pos_splices(i)

        for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
```

```
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1,
node2,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2,
node3,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3,
node4,'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



            end
        end

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i+1,1},'$npC"');
```

```
            else

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i+1,1},'$npC"');

            end
          end

        else

         for j=1:NumB+1

            if j==1 || j==NumB+1

                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

            else
                ColumnTag12=(['1',num2str(j),num2str(i),'1','2']);
                ColumnTag22=(['1',num2str(j),num2str(i),'2','2']);
                ColumnTag32=(['1',num2str(j),num2str(i),'3','2']);
                node1 = ([num2str(j), num2str(i),'7']); %num2str = converts
a numeric array into a character array
                node2 = ([num2str(j), num2str(i),'0','1','2']);
                node3 = ([num2str(j), num2str(i),'0','2','2']);
                node4 = ([num2str(j), num2str(i),'0','3','2']);
```

```
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag12, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag22, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag32, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');



            end
           end

         for j=1:NumB+1

             if j==1 || j==NumB+1

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTEXT{i,1},'$npC"');

             else

                ColumnTag11=(['1',num2str(j),num2str(i),'1','1']);
                ColumnTag21=(['1',num2str(j),num2str(i),'2','1']);
                ColumnTag31=(['1',num2str(j),num2str(i),'3','1']);
                node1 = ([num2str(j), num2str(i+1),'5']); %num2str =
converts a numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'0','1','1']);
                node3 = ([num2str(j), num2str(i+1),'0','2','1']);
                node4 = ([num2str(j), num2str(i+1),'0','3','1']);
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag11, node1, node2,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag21, node2, node3,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
                fprintf(fileID, '%27s %6s %6s %6s %12s %17s %12s %4s\r\n',
'element forceBeamColumn', ColumnTag31, node3, node4,
'$IDColTransf','"$IntegrationC', CSTIN{i,1},'$npC"');
```

```
            end


        end




    end
end



for i=2:NumS+1

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#BEAM ELEMENTS FOR THE FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#command: element elasticBeamColumn $eleID
$iNode $jNode $A $E $I $transfID');
     fprintf(fileID, '%s\r\n', '#eleID convention:  "2xy" where 2 = beams,
x = Bay #, y = Floor #');
     fprintf(fileID, '%s\r\n', '');

     for j=1:NumB

            BeamRBSTag=(['2',num2str(j),num2str(i)]);
            BeamTag1=(['2',num2str(j),num2str(i),'11']);
            BeamTag2=(['2',num2str(j+1),num2str(i),'22']);
            nodeRBS1 = ([num2str(j), num2str(i),'2']); %num2str = converts
a numeric array into a character array
            nodeRBS2 = ([num2str(j+1), num2str(i),'3']);
            nodeT1_1=([num2str(j), num2str(i),'005']);
            nodeT1_2=([num2str(j), num2str(i),'1']);
            nodeT2_1=([num2str(j+1), num2str(i),'4']);
            nodeT2_2=([num2str(j+1), num2str(i),'100']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag1, nodeT1_1, nodeT1_2,
beam_A(i-1),29000, beam_Ix(i-1)*0.9, '$transfTag');
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamRBSTag, nodeRBS1, nodeRBS2,
beam_A(i-1),29000, beam_Ix(i-1)*(n+1)/n*0.9, '$transfTag');
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', BeamTag2, nodeT2_1, nodeT2_2,
beam_A(i-1),29000, beam_Ix(i-1)*0.9, '$transfTag');
     end

end

%TRUSS ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#TRUSS ELEMENTS FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '#command: element truss $eleID $iNode $jNode $A
$materialID');
```

```
fprintf(fileID, '%s\r\n', '#eleID convention: 6xy, 6 = truss link, x = Bay
#, y = Floor #');
fprintf(fileID, '%s\r\n', 'set TrussMatID  600');
fprintf(fileID, '%s\r\n', 'set Arigid  1000.0');
fprintf(fileID, '%s\r\n', 'set Irigid  0.0001');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic   $TrussMatID
29000');


for i=2:NumS+1

     fprintf(fileID, '%s %u\r\n', '','');
     TrussTag=(['6','4',num2str(i)]);
     node1 = (['4', num2str(i),'005']); %num2str = converts a numeric array
into a character array
     node2 = (['5', num2str(i)]);
     fprintf(fileID, '%13s %6s %6s %6s %21s %12s\r\n', 'element truss',
TrussTag, node1, node2, '  $Arigid   $TrussMatID');

end

%LINK ELEMENTS:
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#LEANING COLUMN FOR THE BUILDING:','');
fprintf(fileID, '%s\r\n', '# eleID convention:  7xy, 7 = p-delta columns, x
= Pier #, y = Story #');
fprintf(fileID, '%s\r\n', '');

for i=1:NumS


 %fprintf(fileID, '%s %u\r\n', '','');
 LeaningTag=(['7','5',num2str(i)]);
 node1 = (['5',num2str(i)]); %num2str = converts a numeric array into a
character array
 node2 = (['5',num2str(i+1)]);
 fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e %12s\r\n', 'element
elasticBeamColumn', LeaningTag, node1, node2, 10000000,29000, 0.001,
'$transfTag');


end


%PANEL ZONE ELEMENTS:


for i=2:NumS+1

    for j=1:NumB+1


            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', '#ELEMENTS FOR THE PANEL ZONE
FLOOR:',i,' #PIER: ',j);
```

```
            fprintf(fileID, '%s\r\n', '# eleID convention:  500xya, 500 =
panel zone element, x = Pier #, y = Floor #');
            fprintf(fileID, '%s\r\n', '# "a" convention: defined in
elemPanelZone2D.tcl, but 1 = top left element');
            fprintf(fileID, '%s\r\n', '');

            PanelElemTag1=(['500',num2str(j),num2str(i),'1']);
            node1 = ([num2str(j),num2str(i),'002']); %num2str = converts a
numeric array into a character array
            node2 = ([num2str(j),num2str(i),'7']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag1, node1, node2,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag2=(['500',num2str(j),num2str(i),'2']);
            node3 = ([num2str(j),num2str(i),'7']); %num2str = converts a
numeric array into a character array
            node4 = ([num2str(j),num2str(i),'003']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag2, node3, node4,
10000000,29000, 1000000, '$transfTag');


            PanelElemTag3=(['500',num2str(j),num2str(i),'3']);
            node5 = ([num2str(j),num2str(i),'004']); %num2str = converts a
numeric array into a character array
            node6 = ([num2str(j),num2str(i),'005']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag3, node5, node6,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag4=(['500',num2str(j),num2str(i),'4']);
            node7 = ([num2str(j),num2str(i),'005']); %num2str = converts a
numeric array into a character array
            node8 = ([num2str(j),num2str(i),'006']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag4, node7, node8,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag5=(['500',num2str(j),num2str(i),'5']);
            node9 = ([num2str(j),num2str(i),'007']); %num2str = converts a
numeric array into a character array
            node10 = ([num2str(j),num2str(i),'5']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag5, node9, node10,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag6=(['500',num2str(j),num2str(i),'6']);
            node11 = ([num2str(j),num2str(i),'5']); %num2str = converts a
numeric array into a character array
            node12 = ([num2str(j),num2str(i),'008']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag6, node11, node12,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag7=(['500',num2str(j),num2str(i),'7']);
```

```
            node13 = ([num2str(j),num2str(i),'009']); %num2str = converts a
numeric array into a character array
            node14 = ([num2str(j),num2str(i),'100']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag7, node13, node14,
10000000,29000, 1000000, '$transfTag');

            PanelElemTag8=(['500',num2str(j),num2str(i),'8']);
            node15 = ([num2str(j),num2str(i),'100']); %num2str = converts a
numeric array into a character array
            node16 = ([num2str(j),num2str(i),'001']);
            fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag8, node15, node16,
10000000,29000, 1000000, '$transfTag');


    end
end

fclose(fileID);

end
```

ANEXO C1: FUNCIÓN MATLAB DE NODOS PARA MODELOS L/12 FIBER HINGE

```
function Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,b1,a1)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

fileID = fopen(['Nodes','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# NODES OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');

%COORDINATES OF THE NODES CORRESPONDING TO FIRST FLOOR:

x=0.0;
y=0.0;

n=NumB+2;
m=n*10+1;

%Nodes at the boundaries:
%nodeID convention:  "xy" where x = Pier # and y = Floor #
fprintf(fileID, '%s\r\n', '# FLOOR 1');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes at the boundaries:');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier # and y
= Floor #');
fprintf(fileID, '%s %u\r\n', '#Leaning column node: ',m);
% fprintf(fileID, '%s\r\n', '');

i=1;

for j=1:(NumB+2)

%         fprintf(fileID, '%s\r\n', '');
        nodenum = ([num2str(j), num2str(i)]); %num2str = converts a numeric
array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




x=0.0;
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '11', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '21', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '31', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '41', x+3*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '51', x+4*bay, y);
```

```
%Nodes for the springs at Floor 1:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes for the springs at Floor 1:');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier #, y =
Floor #, "a" convention: 5 = below; 7 = above');
% fprintf(fileID, '%s\r\n', '');

k=7;

for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '117', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '217', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '317', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '417', x+3*bay, y);

%COORDINATES OF THE NODES FOR THE COLUMNS:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

 for i=2:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE COLUMNS AT FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 5 = below; 7 = above');
%      fprintf(fileID, '%s\r\n', '');

     if i==2
            y=h1;
        else
            y=y+hi;
     end
     x=0;
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');
```

```
        k=5;
        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-beam_d(i-1)/2);

        k=7;
        nodenum = ([num2str(j), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+beam_d(i-1)/2);


        x=x+bay;
     end


    %end
 end




%COORDINATES FOR THE NODES FOR THE LEANING COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5,6 = below; 7,8 = above; (used for columns)
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#Nodes for the Leaning Column');
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier #, y =
Floor #');
x=4*bay;
y=h1;
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', num2str(52), x, y);

 for i=3:NumS+1

     nodenum = (['5', num2str(i)]); %num2str = converts a numeric array
into a character array
%      if i==NumS+1;
%         y2=h1+(NumS-1)*hi-beam_d(i-1)/2;
%         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y2);
%
%          break
%      end


     y=y+hi;
     fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x, y);


 end
```

```
%COORDINATES FOR THE NODES FOR THE SPRINGS AT THE SPLICES OF THE COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = story #, a = 91 down 91 up
 y=h1+1.5*hi;
 for i=3:2:NumS

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the splices at story ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a"= 91 inferior node, 92 superior node');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s\r\n', '');
     x=0.0;

     for j=1:NumB+1

         k=91;
         nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         k=k+1;
         nodenum = ([num2str(j), num2str(i), num2str(k)]);
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         x=x+bay;
     end
    y=y+2*hi;
 end



 %COORDINATES FOR THE NODES AT THE PANEL ZONES:


fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xybc" where x = Pier #, y
= Floor #, bc = location relative to beam-column joint');
fprintf(fileID, '%s\r\n', '#"bc" conventions: 001,002 = top left of
joint');
fprintf(fileID, '%s\r\n', '#003,004 = top right of joint');
fprintf(fileID, '%s\r\n', '#005= middle right of joint; (vertical middle,
horizontal right');
fprintf(fileID, '%s\r\n', '#006,007 = btm right of joint');
fprintf(fileID, '%s\r\n', '#008,009 = btm left of joint');
fprintf(fileID, '%s\r\n', '#100= middle left of joint; (vertical middle,
horizontal left');
fprintf(fileID, '%s\r\n', '#note: top center and btm center nodes were
previously defined as xy7 and xy6, respectively, at Floor 2(center =
horizonal center)');

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS
```

```
    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end



 for i=2:NumS+1


fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
     fprintf(fileID, '%s\r\n', '');

%      if i==2
%             b=h1;
%      end

     x=0.0;

     for j=1:NumB+1

if i-1==pos_splices(i-1)

    if pos_splices(i-1)==NumS

         if j==1 || j==NumB+1
           a=extcol_d;

           a=incol_d;
         end
        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
```

```
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;

    else


        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
```

```
        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;
    end

else



%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '');
```

```
     if i==2
            b=h1;
     end

%      x=0.0;

%      for j=1:NumB+1
        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);
```

```
        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');



     end

    x=x+bay;
 end
b=b+hi;

end



%  COORDINATES FOR THE NODES AT THE RBS:
% nodeID convention:  "xy0Za" where x = Bay #, y = Floor #,Z= Position of
discretization from the joint through the beam (1-7) , a = location
relative to beam-column joint
% "a" convention: 4 = left; 3 = right;



for i=2:NumS+1


fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the RBS section of Floor ',
i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy0Za" where x = Bay
#, y = Floor #,Z= Position of discretization from the joint through the
beam (1-7) , "a" convention: 4 = left; 3 = right');
     fprintf(fileID, '%s\r\n', '');

     x1=b1(i-1)/6;

      if i==2
            b=h1;
      else
          b=h1+(i-2)*hi;
      end


       x=0.0;



for j=1:NumB

if i-1==pos_splices(i-1)
     if pos_splices(i-1)==NumS
```

```
          if j==1
          a=extcol_d;
          be=incol_d;
          raz=0;
          for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


         raz=raz+x1;

          end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i-1)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

              if j==NumB

          a=extcol_d;
          be=incol_d;


           raz=0;
          for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
          end
```

```
           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end

              else
            a=incol_d;


            raz=0;
                   for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
              end
          end

     else



          if j==1
           a=extcol_d;
           be=incol_d;
           raz=0;
           for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i)/2)+a1(i-1)+raz, b);
```

```
         raz=raz+x1;

           end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

              if j==NumB

           a=extcol_d;
           be=incol_d;


            raz=0;
           for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
           end

           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end
```

```
            else
            a=incol_d;


            raz=0;
                   for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
               end
          end
     end




else

%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the RBS section of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy0Za" where x =
Bay #, y = Floor #,Z= Position of discretization from the joint through the
beam (1-7) , "a" convention: 4 = left; 3 = right');
%      fprintf(fileID, '%s\r\n', '');

%      x1=b1(i-1)/6;
%
%       if i==2
%             b=h1;
%       else
%           b=h1+(i-2)*hi;
%       end
%
```

```
%       for j=1:NumB

          if j==1
           a=extcol_d;
           be=incol_d;
           raz=0;
           for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


         raz=raz+x1;

           end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i-1)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

               if j==NumB

           a=extcol_d;
           be=incol_d;


            raz=0;
           for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i-1)/2)+a1(i-1)+raz, b);
```

```
        raz=raz+x1;
           end

           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end

               else
            a=incol_d;


            raz=0;
                   for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
               end
          end




      end

  x=x+bay;

end

end
```

```
y=0.0;

for i=1:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE FIBERS IN THE COLUMNS AT
FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "yx0za" where x = Pier
#, y = Floor #, z= position of the fiber counting from the joint (1-3), "a"
convention: 1 = below; 2 = above');
%      fprintf(fileID, '%s\r\n', '');



     if i==1
            y=0;
     else if i==2
             y=h1;
        else
            y=y+hi;
         end
     end
     x=0;
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        if i==1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+(1/36)*h1*(k));

            end
        else
       if i==NumS+1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));

            end
       else
           if i==2

           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));
```

```
            end

            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*h1*(k));

            end

       else
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

            end

            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));
        end
            end
       end
        end


        x=x+bay;
     end

    %end
 end

fclose(fileID);
end
```

ANEXO C2: FUNCIÓN MATLAB DE NODOS PARA MODELOS L/12 FIBER

```
function Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,b1,a1)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

fileID = fopen(['Nodes','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# NODES OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');

%COORDINATES OF THE NODES CORRESPONDING TO FIRST FLOOR:

x=0.0;
y=0.0;

n=NumB+2;
m=n*10+1;

%Nodes at the boundaries:
%nodeID convention:  "xy" where x = Pier # and y = Floor #
fprintf(fileID, '%s\r\n', '# FLOOR 1');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes at the boundaries:');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier # and y
= Floor #');
fprintf(fileID, '%s %u\r\n', '#Leaning column node: ',m);
% fprintf(fileID, '%s\r\n', '');

i=1;

for j=1:(NumB+2)

%         fprintf(fileID, '%s\r\n', '');
        nodenum = ([num2str(j), num2str(i)]); %num2str = converts a numeric
array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




x=0.0;
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '11', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '21', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '31', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '41', x+3*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '51', x+4*bay, y);
```

```
%Nodes for the springs at Floor 1:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes for the springs at Floor 1:');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier #, y =
Floor #, "a" convention: 5 = below; 7 = above');
% fprintf(fileID, '%s\r\n', '');

k=7;

for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '117', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '217', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '317', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '417', x+3*bay, y);

%COORDINATES OF THE NODES FOR THE COLUMNS:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

 for i=2:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE COLUMNS AT FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 5 = below; 7 = above');
%      fprintf(fileID, '%s\r\n', '');

     if i==2
            y=h1;
        else
            y=y+hi;
     end
     x=0;
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');
        k=5;
```

```
        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-beam_d(i-1)/2);

        k=7;
        nodenum = ([num2str(j), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+beam_d(i-1)/2);


        x=x+bay;
     end


    %end
 end




%COORDINATES FOR THE NODES FOR THE LEANING COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5,6 = below; 7,8 = above; (used for columns)
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#Nodes for the Leaning Column');
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier #, y =
Floor #');
x=4*bay;
y=h1;
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', num2str(52), x, y);

 for i=3:NumS+1

     nodenum = (['5', num2str(i)]); %num2str = converts a numeric array
into a character array
%      if i==NumS+1;
%         y2=h1+(NumS-1)*hi-beam_d(i-1)/2;
%         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y2);
%
%          break
%      end


     y=y+hi;
     fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x, y);


 end
```

```
%COORDINATES FOR THE NODES FOR THE SPRINGS AT THE SPLICES OF THE COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = story #, a = 91 down 91 up
 y=h1+1.5*hi;
 for i=3:2:NumS

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the splices at story ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a"= 91 inferior node, 92 superior node');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s\r\n', '');
     x=0.0;

     for j=1:NumB+1

         k=91;
         nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         k=k+1;
         nodenum = ([num2str(j), num2str(i), num2str(k)]);
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         x=x+bay;
     end
    y=y+2*hi;
 end



 %COORDINATES FOR THE NODES AT THE PANEL ZONES:


fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xybc" where x = Pier #, y
= Floor #, bc = location relative to beam-column joint');
fprintf(fileID, '%s\r\n', '#"bc" conventions: 001,002 = top left of
joint');
fprintf(fileID, '%s\r\n', '#003,004 = top right of joint');
fprintf(fileID, '%s\r\n', '#005= middle right of joint; (vertical middle,
horizontal right');
fprintf(fileID, '%s\r\n', '#006,007 = btm right of joint');
fprintf(fileID, '%s\r\n', '#008,009 = btm left of joint');
fprintf(fileID, '%s\r\n', '#100= middle left of joint; (vertical middle,
horizontal left');
fprintf(fileID, '%s\r\n', '#note: top center and btm center nodes were
previously defined as xy7 and xy6, respectively, at Floor 2(center =
horizonal center)');

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS
```

```
    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end



 for i=2:NumS+1


fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
     fprintf(fileID, '%s\r\n', '');

%      if i==2
%             b=h1;
%      end

     x=0.0;

     for j=1:NumB+1

if i-1==pos_splices(i-1)

    if pos_splices(i-1)==NumS

         if j==1 || j==NumB+1
           a=extcol_d;

           a=incol_d;
         end
        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
```

```
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;

    else


        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
```

```
        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;
    end

else



%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '');
```

```
     if i==2
            b=h1;
     end

%      x=0.0;

%      for j=1:NumB+1
        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);
```

```
        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');



     end

    x=x+bay;
 end
b=b+hi;

end



%  COORDINATES FOR THE NODES AT THE RBS:
% nodeID convention:  "xy0Za" where x = Bay #, y = Floor #,Z= Position of
discretization from the joint through the beam (1-7) , a = location
relative to beam-column joint
% "a" convention: 4 = left; 3 = right;



for i=2:NumS+1


fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the RBS section of Floor ',
i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy0Za" where x = Bay
#, y = Floor #,Z= Position of discretization from the joint through the
beam (1-7) , "a" convention: 4 = left; 3 = right');
     fprintf(fileID, '%s\r\n', '');

     x1=b1(i-1)/6;

      if i==2
            b=h1;
      else
          b=h1+(i-2)*hi;
      end


       x=0.0;



for j=1:NumB

if i-1==pos_splices(i-1)
     if pos_splices(i-1)==NumS
```

```
          if j==1
          a=extcol_d;
          be=incol_d;
          raz=0;
          for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


         raz=raz+x1;

          end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i-1)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

              if j==NumB

          a=extcol_d;
          be=incol_d;


           raz=0;
          for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
          end
```

```
           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end

              else
            a=incol_d;


            raz=0;
                   for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
              end
          end

     else



          if j==1
           a=extcol_d;
           be=incol_d;
           raz=0;
           for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i)/2)+a1(i-1)+raz, b);
```

```
         raz=raz+x1;

           end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

              if j==NumB

           a=extcol_d;
           be=incol_d;


            raz=0;
           for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
           end

           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end
```

```
            else
           a=incol_d;


           raz=0;
                  for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                  end

                  raz=0;

                  for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i)/2)-a1(i-1)-raz, b);

           raz=raz+x1;
                  end
              end
          end
     end




else

%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the RBS section of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy0Za" where x =
Bay #, y = Floor #,Z= Position of discretization from the joint through the
beam (1-7) , "a" convention: 4 = left; 3 = right');
%      fprintf(fileID, '%s\r\n', '');

%      x1=b1(i-1)/6;
%
%       if i==2
%             b=h1;
%       else
%           b=h1+(i-2)*hi;
%       end
%
```

```
%       for j=1:NumB

          if j==1
           a=extcol_d;
           be=incol_d;
           raz=0;
           for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


         raz=raz+x1;

           end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i-1)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

               if j==NumB

           a=extcol_d;
           be=incol_d;


            raz=0;
           for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i-1)/2)+a1(i-1)+raz, b);
```

```
        raz=raz+x1;
           end

           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end

               else
            a=incol_d;


            raz=0;
                   for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
               end
          end




      end

  x=x+bay;

end

end
```

```
y=0.0;

for i=1:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE FIBERS IN THE COLUMNS AT
FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "yx0za" where x = Pier
#, y = Floor #, z= position of the fiber counting from the joint (1-3), "a"
convention: 1 = below; 2 = above');
%      fprintf(fileID, '%s\r\n', '');



     if i==1
            y=0;
     else if i==2
             y=h1;
        else
            y=y+hi;
         end
     end
     x=0;
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        if i==1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+(1/36)*h1*(k));

            end
        else
       if i==NumS+1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));

            end
       else
           if i==2

           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));
```

```
            end

            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*h1*(k));

            end

       else
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

            end

            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));
        end
            end
       end
        end


        x=x+bay;
     end


    %end
 end




fclose(fileID);
end
```

ANEXO C3: FUNCIÓN MATLAB DE NODOS PARA MODELOS L/12 FIBER HINGE AND CPH

```
function Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,sh_int,sh_ext)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

fileID = fopen(['Nodes','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# NODES OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');

%COORDINATES OF THE NODES CORRESPONDING TO FIRST FLOOR:

x=0.0;
y=0.0;

n=NumB+2;
m=n*10+1;

%Nodes at the boundaries:
%nodeID convention:  "xy" where x = Pier # and y = Floor #
fprintf(fileID, '%s\r\n', '# FLOOR 1');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes at the boundaries:');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier # and y
= Floor #');
fprintf(fileID, '%s %u\r\n', '#Leaning column node: ',m);
% fprintf(fileID, '%s\r\n', '');

i=1;

for j=1:(NumB+2)

%         fprintf(fileID, '%s\r\n', '');
        nodenum = ([num2str(j), num2str(i)]); %num2str = converts a numeric
array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




x=0.0;
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '11', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '21', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '31', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '41', x+3*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '51', x+4*bay, y);
```

```
%Nodes for the springs at Floor 1:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes for the springs at Floor 1:');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier #, y =
Floor #, "a" convention: 5 = below; 7 = above');
% fprintf(fileID, '%s\r\n', '');

k=7;

for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '117', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '217', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '317', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '417', x+3*bay, y);

%COORDINATES OF THE NODES FOR THE COLUMNS:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

 for i=2:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE COLUMNS AT FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 5 = below; 7 = above');
%      fprintf(fileID, '%s\r\n', '');

     if i==2
            y=h1;
        else
            y=y+hi;
     end
     x=0;
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');
```

```
        k=5;
        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-beam_d(i-1)/2);

        k=7;
        nodenum = ([num2str(j), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+beam_d(i-1)/2);


        x=x+bay;
     end


    %end
 end




%COORDINATES FOR THE NODES FOR THE LEANING COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5,6 = below; 7,8 = above; (used for columns)
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#Nodes for the Leaning Column');
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier #, y =
Floor #');
x=4*bay;
y=h1;
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', num2str(52), x, y);

 for i=3:NumS+1

     nodenum = (['5', num2str(i)]); %num2str = converts a numeric array
into a character array
%      if i==NumS+1;
%         y2=h1+(NumS-1)*hi-beam_d(i-1)/2;
%         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y2);
%
%          break
%      end


     y=y+hi;
     fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x, y);


 end
```

```
%COORDINATES FOR THE NODES FOR THE SPRINGS AT THE SPLICES OF THE COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = story #, a = 91 down 91 up
 y=h1+1.5*hi;
 for i=3:2:NumS

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the splices at story ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a"= 91 inferior node, 92 superior node');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s\r\n', '');
     x=0.0;

     for j=1:NumB+1

         k=91;
         nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         k=k+1;
         nodenum = ([num2str(j), num2str(i), num2str(k)]);
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         x=x+bay;
     end
    y=y+2*hi;
 end



 %COORDINATES FOR THE NODES AT THE PANEL ZONES:


fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xybc" where x = Pier #, y
= Floor #, bc = location relative to beam-column joint');
fprintf(fileID, '%s\r\n', '#"bc" conventions: 001,002 = top left of
joint');
fprintf(fileID, '%s\r\n', '#003,004 = top right of joint');
fprintf(fileID, '%s\r\n', '#005= middle right of joint; (vertical middle,
horizontal right');
fprintf(fileID, '%s\r\n', '#006,007 = btm right of joint');
fprintf(fileID, '%s\r\n', '#008,009 = btm left of joint');
fprintf(fileID, '%s\r\n', '#100= middle left of joint; (vertical middle,
horizontal left');
fprintf(fileID, '%s\r\n', '#note: top center and btm center nodes were
previously defined as xy7 and xy6, respectively, at Floor 2(center =
horizonal center)');

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS
```

```
    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end



 for i=2:NumS+1


fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
     fprintf(fileID, '%s\r\n', '');

%      if i==2
%             b=h1;
%      end

     x=0.0;

     for j=1:NumB+1

if i-1==pos_splices(i-1)

    if pos_splices(i-1)==NumS

         if j==1 || j==NumB+1
           a=extcol_d;

           a=incol_d;
         end
        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
```

```
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;

    else


        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
```

```
        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;
    end

else



%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '');
```

```
     if i==2
            b=h1;
     end

%      x=0.0;

%      for j=1:NumB+1
        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);
```

```
        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');



     end

    x=x+bay;
 end
b=b+hi;

 end

 %COORDINATES FOR THE NODES FOR THE SPRINGS AT THE BEAMS AT RBS:
%nodeID convention:  "xya" where x = Bay #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 1,2 = left; 3,4 = right; (used for beams)
y=h1;
 for i=2:NumS+1;

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the springs at RBS locations
of Floor ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 1,2 = left; 3,4 = right; (used for
beams)');
     fprintf(fileID, '%s\r\n', '');
     x=sh_ext(i-1);

     if i==NumS+1;
         for j=1:NumB;
             y2=h1+(NumS-1)*hi;
             if j==NumB;

                 k=1;
                 nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str
= converts a numeric array into a character array
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, 2*bay+sh_int(i-1), y2);
                 k=k+1;
                 nodenum = ([num2str(j), num2str(i), num2str(k)]);
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, 2*bay+sh_int(i-1), y2);
                 k=k+1;
                 nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, j*bay-sh_ext(i-1), y2);
                 k=k+1;
                 nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, j*bay-sh_ext(i-1), y2);
                 fprintf(fileID, '%s\r\n', '');
```

```
                break
            end



            k=1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, y2);
            k=k+1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, y2);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y2);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y2);
            fprintf(fileID, '%s\r\n', '');
            x=0.0;
            x=j*bay+sh_int(i-1);



        end
        break
    end

    for j=1:NumB;

        if j==NumB;

            k=1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
2*bay+sh_int(i-1), y);
            k=k+1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
2*bay+sh_int(i-1), y);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_ext(i-1), y);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_ext(i-1), y);
            fprintf(fileID, '%s\r\n', '');
```

```
            break
        end

        k=1;
        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
        k=k+1;
        nodenum = ([num2str(j), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
        k=k+1;
        nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y);
        k=k+1;
        nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y);
        fprintf(fileID, '%s\r\n', '');
        x=0.0;
        x=j*bay+sh_int(i-1);



     end

    y=y+hi;

 end


y=0.0;

for i=1:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE FIBERS IN THE COLUMNS AT
FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "yx0za" where x = Pier
#, y = Floor #, z= position of the fiber counting from the joint (1-3), "a"
convention: 1 = below; 2 = above');
%      fprintf(fileID, '%s\r\n', '');



     if i==1
            y=0;
     else if i==2
             y=h1;
        else
            y=y+hi;
         end
     end
     x=0;
```

```
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        if i==1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+(1/36)*h1*(k));

            end
        else
       if i==NumS+1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));

            end
       else
           if i==2

           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

           end

           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*h1*(k));

           end

       else
           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

           end

           for k=1:3
```

```
        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));
        end
           end
       end
        end


        x=x+bay;
     end


    %end
 end

fclose(fileID);
end
```

ANEXO C4: FUNCIÓN MATLAB DE NODOS PARA MODELOS L/12 FIBER AND CPH

```
function Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,sh_int,sh_ext)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

fileID = fopen(['Nodes','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# NODES OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');

%COORDINATES OF THE NODES CORRESPONDING TO FIRST FLOOR:

x=0.0;
y=0.0;

n=NumB+2;
m=n*10+1;

%Nodes at the boundaries:
%nodeID convention:  "xy" where x = Pier # and y = Floor #
fprintf(fileID, '%s\r\n', '# FLOOR 1');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes at the boundaries:');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier # and y
= Floor #');
fprintf(fileID, '%s %u\r\n', '#Leaning column node: ',m);
% fprintf(fileID, '%s\r\n', '');

i=1;

for j=1:(NumB+2)

%         fprintf(fileID, '%s\r\n', '');
        nodenum = ([num2str(j), num2str(i)]); %num2str = converts a numeric
array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




x=0.0;
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '11', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '21', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '31', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '41', x+3*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '51', x+4*bay, y);
```

```
%Nodes for the springs at Floor 1:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#Nodes for the springs at Floor 1:');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier #, y =
Floor #, "a" convention: 5 = below; 7 = above');
% fprintf(fileID, '%s\r\n', '');

k=7;

for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end




% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '117', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '217', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '317', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '417', x+3*bay, y);

%COORDINATES OF THE NODES FOR THE COLUMNS:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

 for i=2:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE COLUMNS AT FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 5 = below; 7 = above');
%      fprintf(fileID, '%s\r\n', '');

     if i==2
            y=h1;
        else
            y=y+hi;
     end
     x=0;
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');
```

```
        k=5;
        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-beam_d(i-1)/2);

        k=7;
        nodenum = ([num2str(j), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+beam_d(i-1)/2);


        x=x+bay;
     end


    %end
 end




%COORDINATES FOR THE NODES FOR THE LEANING COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5,6 = below; 7,8 = above; (used for columns)
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s %u\r\n', '#Nodes for the Leaning Column');
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier #, y =
Floor #');
x=4*bay;
y=h1;
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', num2str(52), x, y);

 for i=3:NumS+1

     nodenum = (['5', num2str(i)]); %num2str = converts a numeric array
into a character array
%      if i==NumS+1;
%         y2=h1+(NumS-1)*hi-beam_d(i-1)/2;
%         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y2);
%
%          break
%      end


     y=y+hi;
     fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x, y);


 end
```

```
%COORDINATES FOR THE NODES FOR THE SPRINGS AT THE SPLICES OF THE COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = story #, a = 91 down 91 up
 y=h1+1.5*hi;
 for i=3:2:NumS

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the splices at story ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a"= 91 inferior node, 92 superior node');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s\r\n', '');
     x=0.0;

     for j=1:NumB+1

         k=91;
         nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         k=k+1;
         nodenum = ([num2str(j), num2str(i), num2str(k)]);
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         x=x+bay;
     end
    y=y+2*hi;
 end
```

```
 %COORDINATES FOR THE NODES AT THE PANEL ZONES:


fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#nodeID convention:  "xybc" where x = Pier #, y
= Floor #, bc = location relative to beam-column joint');
fprintf(fileID, '%s\r\n', '#"bc" conventions: 001,002 = top left of
joint');
fprintf(fileID, '%s\r\n', '#003,004 = top right of joint');
fprintf(fileID, '%s\r\n', '#005= middle right of joint; (vertical middle,
horizontal right');
fprintf(fileID, '%s\r\n', '#006,007 = btm right of joint');
fprintf(fileID, '%s\r\n', '#008,009 = btm left of joint');
fprintf(fileID, '%s\r\n', '#100= middle left of joint; (vertical middle,
horizontal left');
fprintf(fileID, '%s\r\n', '#note: top center and btm center nodes were
previously defined as xy7 and xy6, respectively, at Floor 2(center =
horizonal center)');

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS
```

```
    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end




 for i=2:NumS+1


fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
     fprintf(fileID, '%s\r\n', '');

%      if i==2
%             b=h1;
%      end

     x=0.0;

     for j=1:NumB+1

if i-1==pos_splices(i-1)

    if pos_splices(i-1)==NumS

         if j==1 || j==NumB+1
           a=extcol_d;

           a=incol_d;
         end
        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
```

```
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;

    else


        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
```

```
        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;
    end

else



%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '');
```

```
     if i==2
            b=h1;
     end

%      x=0.0;

%      for j=1:NumB+1
        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);
```

```
        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
        fprintf(fileID, '%s\r\n', '');



     end

    x=x+bay;
 end
b=b+hi;

 end

 %COORDINATES FOR THE NODES FOR THE SPRINGS AT THE BEAMS AT RBS:
%nodeID convention:  "xya" where x = Bay #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 1,2 = left; 3,4 = right; (used for beams)
y=h1;
 for i=2:NumS+1;

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#Nodes for the springs at RBS locations
of Floor ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 1,2 = left; 3,4 = right; (used for
beams)');
     fprintf(fileID, '%s\r\n', '');
     x=sh_ext(i-1);

     if i==NumS+1;
         for j=1:NumB;
             y2=h1+(NumS-1)*hi;
             if j==NumB;

                 k=1;
                 nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str
= converts a numeric array into a character array
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, 2*bay+sh_int(i-1), y2);
                 k=k+1;
                 nodenum = ([num2str(j), num2str(i), num2str(k)]);
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, 2*bay+sh_int(i-1), y2);
                 k=k+1;
                 nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, j*bay-sh_ext(i-1), y2);
                 k=k+1;
                 nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
                 fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ',
nodenum, j*bay-sh_ext(i-1), y2);
                 fprintf(fileID, '%s\r\n', '');
```

```
                break
            end



            k=1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, y2);
            k=k+1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, y2);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y2);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y2);
            fprintf(fileID, '%s\r\n', '');
            x=0.0;
            x=j*bay+sh_int(i-1);



        end
        break
    end

    for j=1:NumB;

        if j==NumB;

            k=1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
2*bay+sh_int(i-1), y);
            k=k+1;
            nodenum = ([num2str(j), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
2*bay+sh_int(i-1), y);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_ext(i-1), y);
            k=k+1;
            nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
            fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_ext(i-1), y);
            fprintf(fileID, '%s\r\n', '');
```

```
            break
        end

        k=1;
        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
        k=k+1;
        nodenum = ([num2str(j), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
        k=k+1;
        nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y);
        k=k+1;
        nodenum = ([num2str(j+1), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
j*bay-sh_int(i-1), y);
        fprintf(fileID, '%s\r\n', '');
        x=0.0;
        x=j*bay+sh_int(i-1);



     end

    y=y+hi;

 end


y=0.0;

for i=1:NumS+1

%      fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE FIBERS IN THE COLUMNS AT
FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "yx0za" where x = Pier
#, y = Floor #, z= position of the fiber counting from the joint (1-3), "a"
convention: 1 = below; 2 = above');
%      fprintf(fileID, '%s\r\n', '');



     if i==1
            y=0;
     else if i==2
             y=h1;
        else
            y=y+hi;
         end
     end
     x=0;
```

```
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        if i==1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+(1/36)*h1*(k));

            end
        else
       if i==NumS+1
            for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));

            end
       else
           if i==2

           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

           end

           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*h1*(k));

           end

       else
           for k=1:3

        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

           end

           for k=1:3
```

```
        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));
        end
           end
       end
        end


        x=x+bay;
     end


    %end
 end




fclose(fileID);
end
```

ANEXO D: EJEMPLO DE LA FUNCIÓN "NODESG" LA CUAL SOLO SIRVE PARA GRAFICAR LA ESTRUCTURA EN EL MATLAB

Esta función solo fue un artificio para poder cargar los datos necesarios en el Matlab y poder graficar las estructuras. Para esto solo hay que poner en comentarios los títulos de Matlab, y que solo se muetsren los datos de los nudos. El ejemplo siguiente muestra como se hizo para el modelo Fiber Hinge de 8 pisos.

```
function Nodesg(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,NumB,b1,a1)

% NumS=8;     %Number of stories of the building
% bay=240;    %Bay length
% h1=180;     %Height of the first story
% hi=156;     %Height of the rest of stories

fileID = fopen(['Nodesg','.tcl'],'w');
% fprintf(fileID, '%s\r\n', '# NODES OF THE BUILDING');
% fprintf(fileID, '%s\r\n', '');

%COORDINATES OF THE NODES CORRESPONDING TO FIRST FLOOR:

x=0.0;
y=0.0;

% n=NumB+2;
% m=n*10+1;

%Nodes at the boundaries:
%nodeID convention:  "xy" where x = Pier # and y = Floor #
% fprintf(fileID, '%s\r\n', '# FLOOR 1');
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '#Nodes at the boundaries:');
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier # and
y = Floor #');
% fprintf(fileID, '%s %u\r\n', '#Leaning column node: ',m);
% fprintf(fileID, '%s\r\n', '');

i=1;

for j=1:NumB+2

%         fprintf(fileID, '%s\r\n', '');
        nodenum = ([num2str(j), num2str(i)]); %num2str = converts a numeric
array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
```

```
        x=x+bay;
end



x=0.0;
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '11', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '21', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '31', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '41', x+3*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '51', x+4*bay, y);

%Nodes for the springs at Floor 1:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '#Nodes for the springs at Floor 1:');
% fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier #, y
= Floor #, "a" convention: 5 = below; 7 = above');
% fprintf(fileID, '%s\r\n', '');

k=7;

for j=1:NumB+1

%         fprintf(fileID, '%s\r\n', '');

        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);

        x=x+bay;
end



% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '117', x, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '217', x+bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '317', x+2*bay, y);
% fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '417', x+3*bay, y);

%COORDINATES OF THE NODES FOR THE COLUMNS:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5 = below; 7 = above; (used for columns)

 for i=2:NumS+1

%      fprintf(fileID, '%s\r\n', '');
```

```
%      fprintf(fileID, '%s %u\r\n', '#NODES FOR THE COLUMNS AT FLOOR: ',
i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 5 = below; 7 = above');
%      fprintf(fileID, '%s\r\n', '');

     if i==2
            y=h1;
        else
            y=y+hi;
     end
     x=0;
     for j=1:NumB+1

%         fprintf(fileID, '%s\r\n', '');
        k=5;
        nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-beam_d(i-1)/2);

        k=7;
        nodenum = ([num2str(j), num2str(i), num2str(k)]);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+beam_d(i-1)/2);


        x=x+bay;
     end


    %end
 end




%COORDINATES FOR THE NODES FOR THE LEANING COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 5,6 = below; 7,8 = above; (used for columns)
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s %u\r\n', '#Nodes for the Leaning Column');
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy" where x = Pier #, y
= Floor #');
x=4*bay;
y=h1;
% fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', num2str(52), x, y);

 for i=3:NumS+1

     nodenum = (['5', num2str(i)]); %num2str = converts a numeric array
into a character array
```

```
%      if i==NumS+1;
%         y2=h1+(NumS-1)*hi-beam_d(i-1)/2;
%         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y2);
%
%          break
%      end


     y=y+hi;
     fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x, y);


 end


%COORDINATES FOR THE NODES FOR THE SPRINGS AT THE SPLICES OF THE COLUMN:
%nodeID convention:  "xya" where x = Pier #, y = story #, a = 91 down 91 up
 y=h1+1.5*hi;
 for i=3:2:NumS

%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the splices at story ', i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a"= 91 inferior node, 92 superior node');
%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s\r\n', '');
     x=0.0;

     for j=1:NumB+1

         k=91;
         nodenum = ([num2str(j), num2str(i), num2str(k)]); %num2str =
converts a numeric array into a character array
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         k=k+1;
         nodenum = ([num2str(j), num2str(i), num2str(k)]);
         fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y);
         x=x+bay;
     end
    y=y+2*hi;
 end




 %COORDINATES FOR THE NODES AT THE PANEL ZONES:


% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '');
% fprintf(fileID, '%s\r\n', '#nodeID convention:  "xybc" where x = Pier #,
y = Floor #, bc = location relative to beam-column joint');
```

```
% fprintf(fileID, '%s\r\n', '#"bc" conventions: 001,002 = top left of
joint');
% fprintf(fileID, '%s\r\n', '#003,004 = top right of joint');
% fprintf(fileID, '%s\r\n', '#005= middle right of joint; (vertical middle,
horizontal right');
% fprintf(fileID, '%s\r\n', '#006,007 = btm right of joint');
% fprintf(fileID, '%s\r\n', '#008,009 = btm left of joint');
% fprintf(fileID, '%s\r\n', '#100= middle left of joint; (vertical middle,
horizontal left');
% fprintf(fileID, '%s\r\n', '#note: top center and btm center nodes were
previously defined as xy7 and xy6, respectively, at Floor 2(center =
horizonal center)');

aa=3;
zz=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if aa>=NumS
        break;
    end
    pos_splices(aa)=aa;

    aa=aa+2;
    zz=zz+1;
end

for i=2:NumS+1


% fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '');

%      if i==2
%             b=h1;
%      end

     x=0.0;

     for j=1:NumB+1

if i-1==pos_splices(i-1)

    if pos_splices(i-1)==NumS

         if j==1 || j==NumB+1
           a=extcol_d;

           a=incol_d;
         end
        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
```

```
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
%         fprintf(fileID, '%s\r\n', '');
%         x=x+bay;



%      b=b+hi;

    else


        if j==1 || j==NumB+1
           a=extcol_d;
```

```
        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');


        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
%         fprintf(fileID, '%s\r\n', '');
%         x=x+bay;
```

```
%      b=b+hi;
    end

else



%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the panel-zone of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '');

     if i==2
          b=h1;
     end

%      x=0.0;

%      for j=1:NumB+1
        if j==1 || j==NumB+1
           a=extcol_d;

        else
           a=incol_d;
        end

        nodenum1 = ([num2str(j), num2str(i), '001']); %num2str = converts a
numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '002']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b+beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');

        nodenum1 = ([num2str(j), num2str(i), '003']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b+beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '004']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b+beam_d(i-1)/2);

        nodenum = ([num2str(j), num2str(i), '005']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum1 = ([num2str(j), num2str(i), '006']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1,
x+a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '007']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2,
x+a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
```

```
        nodenum1 = ([num2str(j), num2str(i), '008']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum1, x-
a(i-1)/2, b-beam_d(i-1)/2);
        nodenum2 = ([num2str(j), num2str(i), '009']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum2, x-
a(i-1)/2, b-beam_d(i-1)/2);
        %fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodenum1,
nodenum2, '1', '2');
        nodenum = ([num2str(j), num2str(i), '100']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x-
a(i-1)/2, b);

        %if i==(NumS+1);
         %   nodenum = ([num2str(j), num2str(i), '7']);
          %  fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x, b+beam_d(i-1)/2);
        %end
%         fprintf(fileID, '%s\r\n', '');



     end

    x=x+bay;
 end
b=b+hi;

end






 %COORDINATES FOR THE NODES AT THE BEAMS:
%nodeID convention:  "xya" where x = Bay #, y = Floor #, a = location
relative to beam-column joint
%"a" convention: 1 = left; 4 = right; (used for beams)

for i=2:NumS+1



%     fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the beams of Floor ', i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 1 = left; 4 = right; (used for beams)');
%      fprintf(fileID, '%s\r\n', '');
%
%



       x=0.0;
```

```
      for j=1:NumB
if i-1==pos_splices(i-1)

    if pos_splices(i-1)==NumS

     if j==1
           a=extcol_d;
           be=incol_d;

         nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-be(i-1)/2, b);

          else

               if j==NumB

           a=extcol_d;
           be=incol_d;

         nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+be(i-1)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-a(i-1)/2, b);


        else
           a=incol_d;
        nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-a(i-1)/2, b);


               end
          end


 else

    if i==2
            b=h1;
      end
```

```
          if j==1
           a=extcol_d;
           be=incol_d;

         nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-be(i)/2, b);

          else

               if j==NumB

           a=extcol_d;
           be=incol_d;

         nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+be(i)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-a(i)/2, b);


        else
           a=incol_d;
        nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-a(i)/2, b);


               end
          end
    end

else

%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the beams of Floor ', i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier
#, y = Floor #, "a" convention: 1 = left; 4 = right; (used for beams)');
%      fprintf(fileID, '%s\r\n', '');

      if i==2
            b=h1;
      end
```

```
%        x=0.0;

%       for j=1:NumB

          if j==1
           a=extcol_d;
           be=incol_d;

         nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-be(i-1)/2, b);

          else

               if j==NumB

           a=extcol_d;
           be=incol_d;

         nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+be(i-1)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-a(i-1)/2, b);


        else
           a=incol_d;
        nodenum = ([num2str(j), num2str(i), '1']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+a(i-1)/2, b);

        nodenum = ([num2str(j+1), num2str(i), '4']);
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-a(i-1)/2, b);


               end
          end



      end



   x=x+bay;
```

```
end


b=b+hi;
end



 %  COORDINATES FOR THE NODES AT THE RBS:
% nodeID convention:  "xy0Za" where x = Bay #, y = Floor #,Z= Position of
discretization from the joint through the beam (1-7) , a = location
relative to beam-column joint
% "a" convention: 4 = left; 3 = right;



for i=2:NumS+1

x=0.0;


% fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the RBS section of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy0Za" where x =
Bay #, y = Floor #,Z= Position of discretization from the joint through the
beam (1-7) , "a" convention: 4 = left; 3 = right');
%      fprintf(fileID, '%s\r\n', '');

     x1=b1(i-1)/6;

      if i==2
            b=h1;
      else
          b=h1+(i-2)*hi;
      end


%        x=0.0;



      for j=1:NumB

if i-1==pos_splices(i-1)

    if pos_splices(i-1)==NumS

           if j==1
           a=extcol_d;
           be=incol_d;
           raz=0;
           for k=1:7
```

```
         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


         raz=raz+x1;

           end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i-1)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

              if j==NumB

           a=extcol_d;
           be=incol_d;


            raz=0;
           for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
           end

           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
```

```
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end

              else
            a=incol_d;


            raz=0;
                   for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
              end
           end

    else

          if j==1
           a=extcol_d;
           be=incol_d;
           raz=0;
           for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i)/2)+a1(i-1)+raz, b);


         raz=raz+x1;

           end
```

```
        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

              if j==NumB

           a=extcol_d;
           be=incol_d;


            raz=0;
           for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
           end

           raz=0;

           for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end

              else
            a=incol_d;


            raz=0;
                   for k=1:7
```

```
        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
               end
          end

    end



else

%      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#Nodes for the RBS section of Floor ',
i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "xy0Za" where x =
Bay #, y = Floor #,Z= Position of discretization from the joint through the
beam (1-7) , "a" convention: 4 = left; 3 = right');
%      fprintf(fileID, '%s\r\n', '');

%      x1=b1(i-1)/6;
%
%       if i==2
%             b=h1;
%       else
%           b=h1+(i-2)*hi;
%       end


%        x=0.0;



%       for j=1:NumB

          if j==1
           a=extcol_d;
           be=incol_d;
           raz=0;
```

```
           for k=1:7


         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


         raz=raz+x1;

           end



        raz=0;
        for k=1:7

        nodenum = ([num2str(j+1), num2str(i), '0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(be(i-1)/2)-a1(i-1)-raz, b);


        raz=raz+x1;

        end




          else

               if j==NumB

           a=extcol_d;
           be=incol_d;


            raz=0;
           for k=1:7



         nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(be(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
           end

           raz=0;

           for k=1:7
```

```
        nodenum = ([num2str(j+1), num2str(i), '0',num2str(k) , '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

        raz=raz+x1;
           end

              else
            a=incol_d;


            raz=0;
                   for k=1:7


        nodenum = ([num2str(j), num2str(i), '0', num2str(k), '4']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+(a(i-1)/2)+a1(i-1)+raz, b);


        raz=raz+x1;
                   end

                   raz=0;

                   for k=1:7

        nodenum = ([num2str(j+1), num2str(i),'0', num2str(k), '3']);
        fprintf(fileID, '%5s %7s %10.3f %10.3f\r\n', 'node ', nodenum,
x+bay-(a(i-1)/2)-a1(i-1)-raz, b);

            raz=raz+x1;
                   end
              end
          end




      end

x=x+bay;


end



end


y=0.0;
```

```
for i=1:NumS+1

% %      fprintf(fileID, '%s\r\n', '');
%      fprintf(fileID, '%s %u\r\n', '#NODES FOR THE FIBERS IN THE COLUMNS
AT FLOOR: ', i);
%      fprintf(fileID, '%s\r\n', '#nodeID convention:  "yx0za" where x =
Pier #, y = Floor #, z= position of the fiber counting from the joint (1-
3), "a" convention: 1 = below; 2 = above');
% %      fprintf(fileID, '%s\r\n', '');



     if i==1
            y=0;
     else if i==2
             y=h1;
        else
            y=y+hi;
         end
     end
     x=0;
     for j=1:(NumB+1)

%         fprintf(fileID, '%s\r\n', '');

        if i==1
            for k=1:3

        nodenum = ([num2str(i), num2str(j), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+(1/36)*h1*(k));

            end
        else
       if i==NumS+1
            for k=1:3

        nodenum = ([num2str(i), num2str(j), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));

            end
       else
           if i==2

           for k=1:3

        nodenum = ([num2str(i), num2str(j), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

           end
```

```
            for k=1:3

        nodenum = ([num2str(i), num2str(j), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*h1*(k));

            end

       else
            for k=1:3

        nodenum = ([num2str(i), num2str(j), '0', num2str(k), '2']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y+((beam_d(i-1))/2)+(1/36)*hi*(k));

            end

            for k=1:3

        nodenum = ([num2str(i), num2str(j), '0', num2str(k), '1']);
%num2str = converts a numeric array into a character array
        fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', nodenum, x,
y-((beam_d(i-1))/2)-(1/36)*hi*(k));
        end
            end
       end
        end


        x=x+bay;
     end


    %end
 end

fclose(fileID);
end
```

ANEXO E: FUNCIÓN DE MATLAB PARA DEFINIR LAS SECCIONES EN LAS VIGAS Y COLUMNAS

Funciones validas para todos los modelos

```
function [BST0, BST1, BST2]=BeamSecTag(beam_sizes,NumS)

nn=NumS;
bs=beam_sizes;

for i=2:nn+1
BST0(i-1,1) = strcat('$',bs(i-1),'aa',num2str(i),'aa','RBS0');
BST1(i-1,1) = strcat('$',bs(i-1),'aa',num2str(i),'aa','RBS1');
BST2(i-1,1)= strcat('$',bs(i-1),'aa',num2str(i),'aa','RBS2');

end


end


function [BSTT0, BSTT1, BSTT2]=BeamSecTag2(beam_sizes,NumS)


nn=NumS;
bs=beam_sizes;

for i=2:nn+1
BSTT0(i-1,1) = strcat(bs(i-1),'aa',num2str(i),'aa','RBS0');
BSTT1(i-1,1) = strcat(bs(i-1),'aa',num2str(i),'aa','RBS1');
BSTT2(i-1,1)= strcat(bs(i-1),'aa',num2str(i),'aa','RBS2');
end
end


function [CSTEXT,CSTIN]=ColSecTag(extcol_sizes,incol_sizes,NumS)

ec=extcol_sizes;
ic=incol_sizes;
nn=NumS;

for i=1:nn
CSTEXT(i,1) = strcat('$',ec(i),'aa',num2str(i),'aa','ext');
CSTIN(i,1) = strcat('$',ic(i),'aa',num2str(i),'aa','in ');

end
end
```

ANEXO F: FUNCIÓN DE MATLAB PARA ESTABLECER LOS CONSTRAINTS

```
function constraints(NumS)

fileID = fopen(['Constraints','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# EQUAL DOF OF THE STRUCTURE');
fprintf(fileID, '%s\r\n', '');

a=3;
z=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;
    z=z+1;

end
num_splices=z;




for i=1:NumS

     if i==pos_splices(i)
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#EQUAL DOF FOR SPLICES IN THE STORY:
', i);
        fprintf(fileID, '%s\r\n', '# Spring ID: "3xya", where 3 = col
spring, x = Pier #, y = Story #, a = location in story');
        fprintf(fileID, '%s\r\n', '# "a" convention: 1 = bottom of story, 2
= top of story, 3=splice');
        fprintf(fileID, '%s\r\n', '');

         for j=1:4

%              ColumnTag2=(['3',num2str(j),num2str(i),'3']);
              node3 = ([num2str(j), num2str(i),'91']); %num2str = converts
a numeric array into a character array
              node4 = ([num2str(j), num2str(i),'92']);
              fprintf(fileID, '%s\r\n', '');
%               fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s
%12s\r\n', 'element zeroLength', ColumnTag2, node3, node4, '-mat ', '666',
'-dir ', '6');
              fprintf(fileID, '%s\r\n', '');
              fprintf(fileID, '%8s %6s %6s %6s %6s %12s\r\n', 'equalDOF',
node3, node4, '1', '2', '3');
              fprintf(fileID, '%s\r\n', '');
```

```
        end

    end

end
```

```
for i=2:NumS+1
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', '#EQUAL DOF FOR THE PANEL ZONE
FLOOR:',i);
%             fprintf(fileID, '%s\r\n', '# eleID convention:  4xy00, 4 =
panel zone spring, x = Pier #, y = Floor #');

    for j=1:4

            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', ' #PIER: ',j);
            fprintf(fileID, '%s\r\n', '');

%             PanelTag=(['4',num2str(j),num2str(i),'00']);
            node1 = ([num2str(j),num2str(i),'003']); %num2str = converts a
numeric array into a character array
            node2 = ([num2str(j),num2str(i),'004']);
%             fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag1, node1, node2,
10000000,29000, 1000000, '$PDeltaTransf;');
%             fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s
%12s\r\n', 'element zeroLength', PanelTag, node1, node2, '-mat ', PanelTag,
'-dir ', '6');
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', node1,
node2, '1', '2');

            node3 = ([num2str(j),num2str(i),'006']); %num2str = converts a
numeric array into a character array
            node4 = ([num2str(j),num2str(i),'007']);
            fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', node3,
node4, '1', '2');
%
            node5 = ([num2str(j),num2str(i),'008']); %num2str = converts a
numeric array into a character array
            node6 = ([num2str(j),num2str(i),'009']);
            fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', node5,
node6, '1', '2');
%
```

```
            node7 = ([num2str(j),num2str(i),'001']); %num2str = converts a
numeric array into a character array
            node8 = ([num2str(j),num2str(i),'002']);
            fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', node7,
node8, '1', '2');



    end
end


fprintf(fileID, '%s\r\n', '# constrain beam-column joints in a floor to
have the same lateral displacement using the "equalDOF" command');
fprintf(fileID, '%s\r\n', '# command: equalDOF $MasterNodeID $SlaveNodeID
$dof1 $dof2... #This command is used to construct a multi-point constraint
between nodes.');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', 'set dof1 1;  # constrain movement in dof 1 (x-
direction)');
fprintf(fileID, '%s\r\n', '');


for i=2:NumS+1
    fprintf(fileID, '%s\r\n', '');

  for j=2:5
    if j==5
        ConsTag1=(['1',num2str(i),'005']);
        ConsTag3=(['5',num2str(i)]);
        fprintf(fileID, '%8s %s %s %13s\r\n', 'equalDOF', ConsTag1,
ConsTag3, '    $dof1;');

    else
        ConsTag1=(['1',num2str(i),'005']);
        ConsTag2=([num2str(j),num2str(i),'005']);
        fprintf(fileID, '%8s %s %s %13s\r\n', 'equalDOF', ConsTag1,
ConsTag2, '$dof1;');
    end
  end
end


fclose(fileID);
end
```

ANEXO G: FUNCIÓN DE MATLAB PARA ESTABLECER LAS MASAS

```
function masses(NumS)

fileID = fopen(['Masses','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# CALCULATE NODAL MASSES -- LUMP FLOOR MASSES AT
FRAME NODES');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', 'set g 386.2;  # acceleration due to gravity');
fprintf(fileID, '%s\r\n', 'set Negligible 1e-9');
fprintf(fileID, '%s\r\n', 'set Floor2Weight  800.45;  # weight of Floor 2
in kips');

W1=800.45;
Wi=796.70;
Wtop=720.38;

Wtotal=W1+Wtop+Wi*(NumS-2);

for i=3:NumS+1

   if i==NumS+1

       Tag1=(['set Floor',num2str(i),'Weight 720.38;  # weight of Floor in
kips']);
       fprintf(fileID, '%s\r\n', Tag1);
       break
   end

    Tag2=(['set Floor',num2str(i),'Weight  796.70;  # weight of Floor in
kips']);
    fprintf(fileID, '%s\r\n', Tag2);

end

for i=2:NumS+1

    if i==2
        floor_weight=W1;
    else
        if i==NumS+1
            floor_weight=Wtop;
        else
            floor_weight=Wi;
        end
    end


    interior(i-1)=floor_weight*0.20/(386.4);
    exterior(i-1)=floor_weight*0.30/(386.4);

end

fprintf(fileID, '%s\r\n', '');
```

```
for i=2:NumS+1
    for j=1:4
        if j==1 || j==4
            MassTag=([num2str(j),num2str(i),'005']);
            fprintf(fileID, '%4s %s %4s %13s %13s\r\n', 'mass', MassTag,
num2str(exterior(i-1)),' $Negligible',' $Negligible;');
        else
            MassTag=([num2str(j),num2str(i),'005']);
            fprintf(fileID, '%4s %s %4s %13s %13s\r\n', 'mass', MassTag,
num2str(interior(i-1)),' $Negligible',' $Negligible;');
        end
    end
    fprintf(fileID, '%s\r\n', '');
end


end
```

ANEXO H: FUNCIÓN DE MATLAB PARA ESTABLECER LAS SECCIONES

```
function
Sections(beam_sizes,NumS,extcol_sizes,incol_sizes,beam_d,beam_bf,beam_tf,be
am_tw,extcol_d,extcol_bf,extcol_tf,extcol_tw,incol_d,incol_bf,incol_tf,inco
l_tw)

bf=beam_bf;
d=beam_d;
tw=beam_tw;
tf=beam_tf;

for i=1:NumS

R(i)=(1/8)*(bf(i)+(9/4)*(d(i)/bf(i))*d(i));
end


for i=1:NumS
bfeff0(i)=2*(R(i)+0.25*bf(i)-sqrt((R(i)*R(i))-((0.75*d(i)*(0/6))^2)));
bfeff1(i)=2*(R(i)+0.25*bf(i)-sqrt((R(i)*R(i))-((0.75*d(i)*(1/6))^2)));
bfeff2(i)=2*(R(i)+0.25*bf(i)-sqrt((R(i)*R(i))-((0.75*d(i)*(2/6))^2)));




end




[BST0, BST1, BST2]=BeamSecTag(beam_sizes,NumS);
[BSTT0, BSTT1, BSTT2]=BeamSecTag2(beam_sizes,NumS);
[CSTEXT,CSTIN]=ColSecTag(extcol_sizes,incol_sizes,NumS);

Beam_K0=[1.41,1.50,1.50,1.34,1.34,1.34,1.27,1.19];
Beam_K1=[1.125,1.125,1.125,1.0625,1.0625,1.0625,1.0625,0.875];

Colext_K0=[1.46,1.46,1.46,1.46,1.46,1.46,1.46,1.38];
Colext_K1=[1.125,1.125,1.125,1.125,1.125,1.125,1.125,1.0625];

Colin_K0=[1.72,1.72,1.72,1.72,1.72,1.46,1.46,1.38];
Colin_K1=[1.1875,1.1875,1.1875,1.1875,1.1875,1.125,1.125,1.0625];




fileID = fopen(['Sections','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# SECTIONS OF THE BUILDING');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# Beam Section Tag');
```

```
for i=1:NumS

fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT0{i,1}, num2str(i+10));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'bfeff0', num2str(bfeff0(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'd0', num2str(d(i,1)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tw0', num2str(tw(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tf0', num2str(tf(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K0', num2str(Beam_K0(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K1', num2str(Beam_K1(i)));
fprintf(fileID, '%s %15s %s %s %s %s %s %s %s\r\n',
'CreateWSection',BST0{i,1},'$matTagB','$d0',
'$bfeff0','$tw0','$tf0','$K0','$K1');
fprintf(fileID, '%s\r\n', '');


fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT1{i,1}, num2str(i+20));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'bfeff1', num2str(bfeff1(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'd1', num2str(d(i,1)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tw1', num2str(tw(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tf1', num2str(tf(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K0', num2str(Beam_K0(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K1', num2str(Beam_K1(i)));
fprintf(fileID, '%s %15s %s %s %s %s %s %s %s\r\n',
'CreateWSection',BST1{i,1},'$matTagB','$d1',
'$bfeff1','$tw1','$tf1','$K0','$K1');
fprintf(fileID, '%s\r\n', '');


fprintf(fileID, '%s %20s %5s\r\n', 'set', BSTT2{i,1}, num2str(i+30));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'bfeff2', num2str(bfeff2(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'd2', num2str(d(i,1)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tw2', num2str(tw(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tf2', num2str(tf(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K0', num2str(Beam_K0(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K1', num2str(Beam_K1(i)));
fprintf(fileID, '%s %15s %s %s %s %s %s %s %s\r\n',
'CreateWSection',BST2{i,1},'$matTagB','$d2',
'$bfeff2','$tw2','$tf2','$K0','$K1');
fprintf(fileID, '%s\r\n', '');




end

fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# Column Section Tag');

for i=1:NumS
```

```
fprintf(fileID, '%s %17s %5s \r\n', 'set',
strcat(extcol_sizes{i,1},'aa',num2str(i),'aa','ext'), num2str(i+100));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'bf', num2str(extcol_bf(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'd', num2str(extcol_d(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tw', num2str(extcol_tw(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tf', num2str(extcol_tf(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K0', num2str(Colext_K0(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K1', num2str(Colext_K1(i)));
fprintf(fileID, '%s %15s %s %s %s %s %s %s %s\r\n',
'CreateWSection',CSTEXT{i,1},'$matTagC','$d',
'$bf','$tw','$tf','$K0','$K1');
fprintf(fileID, '%s\r\n', '');



fprintf(fileID, '%s %17s %5s\r\n', 'set',
strcat(incol_sizes{i,1},'aa',num2str(i),'aa','in '), num2str(i+200));

fprintf(fileID, '%s %s %5s\r\n', 'set', 'bf', num2str(extcol_bf(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'd', num2str(extcol_d(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tw', num2str(extcol_tw(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'tf', num2str(extcol_tf(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K0', num2str(Colin_K0(i)));
fprintf(fileID, '%s %s %5s\r\n', 'set', 'K1', num2str(Colin_K1(i)));
fprintf(fileID, '%s %15s %s %s %s %s %s %s %s\r\n',
'CreateWSection',CSTIN{i,1},'$matTagC','$d',
'$bf','$tw','$tf','$K0','$K1');
fprintf(fileID, '%s\r\n', '');


end
fclose(fileID);
end
```

ANEXO I1: FUNCIÓN DE MATLAB PARA ESTABLECER EL MATERIAL STEEL02 EN FIBER HINGE Y FIBER

```
function
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,inc
ol_bf,incol_tf,beam_d,ext_dblplate,int_dblplate)


fileID = fopen(['Material','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# PROPERTIES FOR THE FIBERS');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# ASTM A992');
fprintf(fileID, '%s\r\n', 'set Fy [expr 55.*1.1]');
fprintf(fileID, '%s\r\n', 'set E 29000');
fprintf(fileID, '%s\r\n', 'set b [expr 0.005]');
fprintf(fileID, '%s\r\n', 'set esh [expr 0.015]');
fprintf(fileID, '%s\r\n', 'set bd [expr 0.010]');
fprintf(fileID, '%s\r\n', 'set Res [expr 0.500]');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial Steel02  10101 [expr $Fy] [expr
$E] [expr $b] 20 0.925 0.15');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', 'set matTagB 10101');
fprintf(fileID, '%s\r\n', 'set matTagC 10101');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# Bases');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    666   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    331   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    332   10e+8');
fprintf(fileID, '%s\r\n', '');

%COLUMN SPRINGS:

a=3;
z=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;
    z=z+1;

end
num_splices=z;


%PANEL ZONE ELEMENTS:

alpha = 0.01;
%Fy=Fy/1.1;
```

```
for i=1:NumS %2:NumS+1;


 fprintf(fileID, '%s\r\n', '');
 fprintf(fileID, '%s\r\n', '');
 fprintf(fileID, '%s %u\r\n', '#MATERIAL PROPERTIES FOR THE SPRINGS OF THE
PANEL ZONE AT FLOOR: ',i+1);


 if i==pos_splices(i)

    for j=1:4

        if j==1 || j==4
              tp=extcol_tw(i+1)+ext_dblplate(i);
              Vy=0.55*Fy*extcol_d(i+1)*tp;
              G= 29000/(2.0 * (1.0 + 0.30));
              gamma_y=Fy/(sqrt(3))/G;
              Ke=Vy/gamma_y;

              gamma_p = 4*gamma_y;
              Kp =
Ke*(extcol_bf(i+1)*extcol_tf(i+1)^2)/(beam_d(i+1)*extcol_d(i+1)*tp); % Eq
3-10

              Ke1 = beam_d(i+1)*(Ke - Kp); % Fig 3-17
              Kp1 = 0.0;
              Ke2 = beam_d(i+1)*Kp; % Fig 3-17
              Kp2 = alpha*beam_d(i+1)*Ke;


              a=num2str(gamma_y*Ke1);
              b=num2str(Ke1);
              c=num2str(Kp1/Ke1);
              d=num2str(gamma_p*Ke2);
              e=num2str(Ke2);
              f=num2str(Kp2/Ke2);

              fprintf(fileID, '%s\r\n', '');
              PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
              PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
              PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
              fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01', PanelTag1, a, b, c);
              fprintf(fileID, '%s\r\n', '');
              fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d, e, f);
              fprintf(fileID, '%s\r\n', '');
              fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);

        else

              Vy=(0.55*Fy*incol_d(i+1)*(incol_tw(i+1)+int_dblplate(i+1)));
              G= 29000/(2.0 * (1.0 + 0.30));
              gamma_y=Fy/(sqrt(3))/G;
```

```
                Ke=Vy/gamma_y;
                tp=incol_tw(i+1)+int_dblplate(i);
                gamma_p = 4*gamma_y;
                Kp =
Ke*(incol_bf(i+1)*incol_tf(i+1)^2)/(beam_d(i+1)*incol_d(i+1)*tp); % Eq 3-10

                Ke1 = beam_d(i+1)*(Ke - Kp); % Fig 3-17
                Kp1 = 0.0;
                Ke2 = beam_d(i+1)*Kp; % Fig 3-17
                Kp2 = alpha*beam_d(i+1)*Ke;

                a1=num2str(gamma_y*Ke1);
                b1=num2str(Ke1);
                c1=num2str(Kp1/Ke1);
                d1=num2str(gamma_p*Ke2);
                e1=num2str(Ke2);
                f1=num2str(Kp2/Ke2);

                fprintf(fileID, '%s\r\n', '');
                PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
                PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
                PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag1, a1, b1, c1);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d1, e1, f1);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);

        end

    end

 else

    for j=1:4;

        if j==1 || j==4
                tp=extcol_tw(i)+ext_dblplate(i);
                Vy=0.55*Fy*extcol_d(i)*tp;
                G= 29000/(2.0 * (1.0 + 0.30));
                gamma_y=Fy/(sqrt(3))/G;
                Ke=Vy/gamma_y;

                gamma_p = 4*gamma_y;
                Kp =
Ke*(extcol_bf(i)*extcol_tf(i)^2)/(beam_d(i)*extcol_d(i)*tp); % Eq 3-10

                Ke1 = beam_d(i)*(Ke - Kp); % Fig 3-17
                Kp1 = 0.0;
                Ke2 = beam_d(i)*Kp; % Fig 3-17
                Kp2 = alpha*beam_d(i)*Ke;
                a=num2str(gamma_y*Ke1);
                b=num2str(Ke1);
                c=num2str(Kp1/Ke1);
```

```
                d=num2str(gamma_p*Ke2);
                e=num2str(Ke2);
                f=num2str(Kp2/Ke2);

                fprintf(fileID, '%s\r\n', '');
                PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
                PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
                PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag1, a, b, c);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d, e, f);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);
        else
                Vy=(0.55*Fy*incol_d(i)*(incol_tw(i)+int_dblplate(i)));
                G= 29000/(2.0 * (1.0 + 0.30));
                gamma_y=Fy/(sqrt(3))/G;
                Ke=Vy/gamma_y;
                tp=incol_tw(i)+int_dblplate(i);
                gamma_p = 4*gamma_y;
                Kp =
Ke*(incol_bf(i)*incol_tf(i)^2)/(beam_d(i)*incol_d(i)*tp); % Eq 3-10
                Ke1 = beam_d(i)*(Ke - Kp); % Fig 3-17
                Kp1 = 0.0;
                Ke2 = beam_d(i)*Kp; % Fig 3-17
                Kp2 = alpha*beam_d(i)*Ke;
                a1=num2str(gamma_y*Ke1);
                b1=num2str(Ke1);
                c1=num2str(Kp1/Ke1);
                d1=num2str(gamma_p*Ke2);
                e1=num2str(Ke2);
                f1=num2str(Kp2/Ke2);
                fprintf(fileID, '%s\r\n', '');
                PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
                PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
                PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag1, a1, b1, c1);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d1, e1, f1);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);

        end
    end
 end
end
fclose(fileID);
Fy=Fy*1.1;
end
```

ANEXO I2: FUNCIÓN DE MATLAB PARA ESTABLECER EL MATERIAL STEEL02 EN "FIBER HINGE AND CPH" Y "FIBER AND CPH"

```
function
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,inc
ol_bf,incol_tf,beam_d,ext_dblplate,int_dblplate,Ke_beam,ass_beam,My_RBS,bea
m_lambda,beam_theta_p,beam_theta_pc)

fileID = fopen(['Material','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# PROPERTIES FOR THE FIBERS');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# ASTM A992');
fprintf(fileID, '%s\r\n', 'set Fy [expr 55.*1.1]');
fprintf(fileID, '%s\r\n', 'set E 29000');
fprintf(fileID, '%s\r\n', 'set b [expr 0.005]');
fprintf(fileID, '%s\r\n', 'set esh [expr 0.015]');
fprintf(fileID, '%s\r\n', 'set bd [expr 0.010]');
fprintf(fileID, '%s\r\n', 'set Res [expr 0.500]');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial Steel02  10101 [expr $Fy] [expr
$E] [expr $b] 20 0.925 0.15');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', 'set matTagB 10101');
fprintf(fileID, '%s\r\n', 'set matTagC 10101');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# Bases');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    666   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    331   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    332   10e+8');
fprintf(fileID, '%s\r\n', '');

%COLUMN SPRINGS:

a=3;
z=0;
pos_splices=zeros(1,NumS);
for i=1:NumS

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;
    z=z+1;

end
num_splices=z;



%BEAM ELEMENTS:

for i=2:NumS+1;
```

```
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#MATERIAL PROPERTIES FOR BEAMS SPRINGS
AT FLOOR:  ', i);
     fprintf(fileID, '%s\r\n', '');

     BeamRBSTag1=(['4','1',num2str(i),'2']);
     ke = num2str(Ke_beam(i-1));
     as = num2str(ass_beam(i-1));
     My= num2str(My_RBS(i-1));
     My2= num2str(-My_RBS(i-1));
     Lamda= num2str(beam_lambda(i-1));
     Cc=num2str(1);
     theta_p=num2str(beam_theta_p(i-1));
     theta_pc=num2str(beam_theta_pc(i-1));
     res=num2str(0.4);
     theta_u=num2str(0.2);
     D=num2str(1);

     fprintf(fileID, '%24s %6s %3s %3s %3s %3s %3s %3s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial
Bilin ', BeamRBSTag1, ke, as, as, My, My2, Lamda, Lamda, Lamda, Lamda, Cc,
Cc, Cc, Cc, theta_p,theta_p,theta_pc,theta_pc, res, res, theta_u, theta_u,
D, D);
     fprintf(fileID, '%s\r\n', '');


     BeamRBSTag2=(['4','2',num2str(i),'1']);
     ke = num2str(Ke_beam(i-1));
     as = num2str(ass_beam(i-1));
     My= num2str(My_RBS(i-1));
     My2= num2str(-My_RBS(i-1));
     Lamda= num2str(beam_lambda(i-1));
     Cc=num2str(1);
     theta_p=num2str(beam_theta_p(i-1));
     theta_pc=num2str(beam_theta_pc(i-1));
     res=num2str(0.4);
     theta_u=num2str(0.2);
     D=num2str(1);

     fprintf(fileID, '%24s %6s %3s %3s %3s %3s %3s %3s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial
Bilin ', BeamRBSTag2, ke, as, as, My, My2, Lamda, Lamda, Lamda, Lamda, Cc,
Cc, Cc, Cc, theta_p,theta_p,theta_pc,theta_pc, res, res, theta_u, theta_u,
D, D);
     fprintf(fileID, '%s\r\n', '');


     BeamRBSTag3=(['4','2',num2str(i),'2']);
     ke = num2str(Ke_beam(i-1));
     as = num2str(ass_beam(i-1));
     My= num2str(My_RBS(i-1));
     My2= num2str(-My_RBS(i-1));
     Lamda= num2str(beam_lambda(i-1));
     Cc=num2str(1);
     theta_p=num2str(beam_theta_p(i-1));
     theta_pc=num2str(beam_theta_pc(i-1));
```

```
     res=num2str(0.4);
     theta_u=num2str(0.2);
     D=num2str(1);

     fprintf(fileID, '%24s %6s %3s %3s %3s %3s %3s %3s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial
Bilin ', BeamRBSTag3, ke, as, as, My, My2, Lamda, Lamda, Lamda, Lamda, Cc,
Cc, Cc, Cc, theta_p,theta_p,theta_pc,theta_pc, res, res, theta_u, theta_u,
D, D);
     fprintf(fileID, '%s\r\n', '');


     BeamRBSTag4=(['4','3',num2str(i),'1']);
     ke = num2str(Ke_beam(i-1));
     as = num2str(ass_beam(i-1));
     My= num2str(My_RBS(i-1));
     My2= num2str(-My_RBS(i-1));
     Lamda= num2str(beam_lambda(i-1));
     Cc=num2str(1);
     theta_p=num2str(beam_theta_p(i-1));
     theta_pc=num2str(beam_theta_pc(i-1));
     res=num2str(0.4);
     theta_u=num2str(0.2);
     D=num2str(1);

     fprintf(fileID, '%24s %6s %3s %3s %3s %3s %3s %3s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial
Bilin ', BeamRBSTag4, ke, as, as, My, My2, Lamda, Lamda, Lamda, Lamda, Cc,
Cc, Cc, Cc, theta_p,theta_p,theta_pc,theta_pc, res, res, theta_u, theta_u,
D, D);
     fprintf(fileID, '%s\r\n', '');


     BeamRBSTag5=(['4','3',num2str(i),'2']);
     ke = num2str(Ke_beam(i-1));
     as = num2str(ass_beam(i-1));
     My= num2str(My_RBS(i-1));
     My2= num2str(-My_RBS(i-1));
     Lamda= num2str(beam_lambda(i-1));
     Cc=num2str(1);
     theta_p=num2str(beam_theta_p(i-1));
     theta_pc=num2str(beam_theta_pc(i-1));
     res=num2str(0.4);
     theta_u=num2str(0.2);
     D=num2str(1);

     fprintf(fileID, '%24s %6s %3s %3s %3s %3s %3s %3s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial
Bilin ', BeamRBSTag5, ke, as, as, My, My2, Lamda, Lamda, Lamda, Lamda, Cc,
Cc, Cc, Cc, theta_p,theta_p,theta_pc,theta_pc, res, res, theta_u, theta_u,
D, D);
     fprintf(fileID, '%s\r\n', '');


     BeamRBSTag6=(['4','4',num2str(i),'1']);
     ke = num2str(Ke_beam(i-1));
     as = num2str(ass_beam(i-1));
```

```
     My= num2str(My_RBS(i-1));
     My2= num2str(-My_RBS(i-1));
     Lamda= num2str(beam_lambda(i-1));
     Cc=num2str(1);
     theta_p=num2str(beam_theta_p(i-1));
     theta_pc=num2str(beam_theta_pc(i-1));
     res=num2str(0.4);
     theta_u=num2str(0.2);
     D=num2str(1);

     fprintf(fileID, '%24s %6s %3s %3s %3s %3s %3s %3s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial
Bilin ', BeamRBSTag6, ke, as, as, My, My2, Lamda, Lamda, Lamda, Lamda, Cc,
Cc, Cc, Cc, theta_p,theta_p,theta_pc,theta_pc, res, res, theta_u, theta_u,
D, D);
     fprintf(fileID, '%s\r\n', '');



end

%PANEL ZONE ELEMENTS:

alpha = 0.01;
%Fy=Fy/1.1;
for i=1:NumS %2:NumS+1;


 fprintf(fileID, '%s\r\n', '');
 fprintf(fileID, '%s\r\n', '');
 fprintf(fileID, '%s %u\r\n', '#MATERIAL PROPERTIES FOR THE SPRINGS OF THE
PANEL ZONE AT FLOOR: ',i+1);


 if i==pos_splices(i)

    for j=1:4

        if j==1 || j==4
              tp=extcol_tw(i+1)+ext_dblplate(i);
              Vy=0.55*Fy*extcol_d(i+1)*tp;
              G= 29000/(2.0 * (1.0 + 0.30));
              gamma_y=Fy/(sqrt(3))/G;
              Ke=Vy/gamma_y;

              gamma_p = 4*gamma_y;
              Kp =
Ke*(extcol_bf(i+1)*extcol_tf(i+1)^2)/(beam_d(i+1)*extcol_d(i+1)*tp); % Eq
3-10

              Ke1 = beam_d(i+1)*(Ke - Kp); % Fig 3-17
              Kp1 = 0.0;
              Ke2 = beam_d(i+1)*Kp; % Fig 3-17
              Kp2 = alpha*beam_d(i+1)*Ke;
```

```
            a=num2str(gamma_y*Ke1);
            b=num2str(Ke1);
            c=num2str(Kp1/Ke1);
            d=num2str(gamma_p*Ke2);
            e=num2str(Ke2);
            f=num2str(Kp2/Ke2);

            fprintf(fileID, '%s\r\n', '');
            PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
            PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
            PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
            fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01', PanelTag1, a, b, c);
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d, e, f);
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);

        else

            Vy=(0.55*Fy*incol_d(i+1)*(incol_tw(i+1)+int_dblplate(i+1)));
            G= 29000/(2.0 * (1.0 + 0.30));
            gamma_y=Fy/(sqrt(3))/G;
            Ke=Vy/gamma_y;
            tp=incol_tw(i+1)+int_dblplate(i);
            gamma_p = 4*gamma_y;
            Kp =
Ke*(incol_bf(i+1)*incol_tf(i+1)^2)/(beam_d(i+1)*incol_d(i+1)*tp); % Eq 3-10

            Ke1 = beam_d(i+1)*(Ke - Kp); % Fig 3-17
            Kp1 = 0.0;
            Ke2 = beam_d(i+1)*Kp; % Fig 3-17
            Kp2 = alpha*beam_d(i+1)*Ke;

            a1=num2str(gamma_y*Ke1);
            b1=num2str(Ke1);
            c1=num2str(Kp1/Ke1);
            d1=num2str(gamma_p*Ke2);
            e1=num2str(Ke2);
            f1=num2str(Kp2/Ke2);

            fprintf(fileID, '%s\r\n', '');
            PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
            PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
            PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
            fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag1, a1, b1, c1);
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d1, e1, f1);
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);

        end
```

```
    end

 else

    for j=1:4;

        if j==1 || j==4
              tp=extcol_tw(i)+ext_dblplate(i);
              Vy=0.55*Fy*extcol_d(i)*tp;
              G= 29000/(2.0 * (1.0 + 0.30));
              gamma_y=Fy/(sqrt(3))/G;
              Ke=Vy/gamma_y;

              gamma_p = 4*gamma_y;
              Kp =
Ke*(extcol_bf(i)*extcol_tf(i)^2)/(beam_d(i)*extcol_d(i)*tp); % Eq 3-10

              Ke1 = beam_d(i)*(Ke - Kp); % Fig 3-17
              Kp1 = 0.0;
              Ke2 = beam_d(i)*Kp; % Fig 3-17
              Kp2 = alpha*beam_d(i)*Ke;


              a=num2str(gamma_y*Ke1);
              b=num2str(Ke1);
              c=num2str(Kp1/Ke1);
              d=num2str(gamma_p*Ke2);
              e=num2str(Ke2);
              f=num2str(Kp2/Ke2);

              fprintf(fileID, '%s\r\n', '');
              PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
              PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
              PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
              fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag1, a, b, c);
              fprintf(fileID, '%s\r\n', '');
              fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d, e, f);
              fprintf(fileID, '%s\r\n', '');
              fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);

        else

              Vy=(0.55*Fy*incol_d(i)*(incol_tw(i)+int_dblplate(i)));
              G= 29000/(2.0 * (1.0 + 0.30));
              gamma_y=Fy/(sqrt(3))/G;
              Ke=Vy/gamma_y;
              tp=incol_tw(i)+int_dblplate(i);
              gamma_p = 4*gamma_y;
              Kp =
Ke*(incol_bf(i)*incol_tf(i)^2)/(beam_d(i)*incol_d(i)*tp); % Eq 3-10

              Ke1 = beam_d(i)*(Ke - Kp); % Fig 3-17
              Kp1 = 0.0;
```

```
                Ke2 = beam_d(i)*Kp; % Fig 3-17
                Kp2 = alpha*beam_d(i)*Ke;

                a1=num2str(gamma_y*Ke1);
                b1=num2str(Ke1);
                c1=num2str(Kp1/Ke1);
                d1=num2str(gamma_p*Ke2);
                e1=num2str(Ke2);
                f1=num2str(Kp2/Ke2);

                fprintf(fileID, '%s\r\n', '');
                PanelTag1=(['4',num2str(j),num2str(i+1),'001']);
                PanelTag2=(['4',num2str(j),num2str(i+1),'002']);
                PanelTag3=(['4',num2str(j),num2str(i+1),'00']);
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag1, a1, b1, c1);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%26s %3s %3s %3s %3s', 'uniaxialMaterial
Steel01 ', PanelTag2, d1, e1, f1);
                fprintf(fileID, '%s\r\n', '');
                fprintf(fileID, '%27s %3s %3s %3s %3s', 'uniaxialMaterial
Parallel ', PanelTag3, PanelTag1, PanelTag2);

        end

    end
 end
end
fclose(fileID);
Fy=Fy*1.1;
end
```

ANEXO I3: MATERIAL HYSTERETIC

Para cambiar de material steel02, solo hay que cambiar la primera parte de la función de materiales (Springs)

```
fileID = fopen(['Material','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# PROPERTIES FOR THE FIBERS');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# Hysteretic');

fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', 'set F1   60.5;');
fprintf(fileID, '%s\r\n', 'set F2   71.5');
fprintf(fileID, '%s\r\n', 'set F3   22');

fprintf(fileID, '%s\r\n', 'set e1   0.0017');
fprintf(fileID, '%s\r\n', 'set e2   0.21');
fprintf(fileID, '%s\r\n', 'set e3   1.385');

fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial Hysteretic 10101 $F1 $e1 $F2
$e2 $F3 $e3 -$F1 -$e1 -$F2 -$e2 -$F3 -$e3 0.5 0.75 0.0 0.0');
fprintf(fileID, '%s\r\n', '');


fprintf(fileID, '%s\r\n', 'set matTagB 10101');
fprintf(fileID, '%s\r\n', 'set matTagC 10101');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# Bases');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    666   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    331   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    332   10e+8');
fprintf(fileID, '%s\r\n', '');
```

ANEXO I4: MATERIAL PARALLEL

Para cambiar de material steel02, solo hay que cambiar la primera parte de la función de materiales (Springs). Para los modelos Fiber Hinge y Fiber se usó el material steel02 en las vigas. Sin embargo, en modelos Fiber Hinge and CPH y Fiber and CPH, no es necesario definirlo, por lo que se puede dejar el código igual, y no se verán afectados los resultados.

```
fileID = fopen(['Material','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# PROPERTIES FOR THE FIBERS');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# ASTM A992');
fprintf(fileID, '%s\r\n', 'set Fy [expr 55.*1.1]');
fprintf(fileID, '%s\r\n', 'set E 29000');
fprintf(fileID, '%s\r\n', 'set b [expr 0.005]');
fprintf(fileID, '%s\r\n', 'set esh [expr 0.015]');
fprintf(fileID, '%s\r\n', 'set bd [expr 0.010]');
fprintf(fileID, '%s\r\n', 'set Res [expr 0.500]');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial Steel02  10101 [expr $Fy] [expr
$E] [expr $b] 20 0.925 0.15');
fprintf(fileID, '%s\r\n', '');

fprintf(fileID, '%s\r\n', 'set matTagB 10101');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', 'set F1   [expr $Fy*(1-$Res)];');
fprintf(fileID, '%s\r\n', 'set F2   [expr $F1+1*$esh*$b*$E];');
fprintf(fileID, '%s\r\n', 'set F3   [expr $F2];');
fprintf(fileID, '%s\r\n', 'set F3n [expr 0.01];');
fprintf(fileID, '%s\r\n', 'set e1   [expr $Fy/$E];');
fprintf(fileID, '%s\r\n', 'set e2   [expr $e1+1*$esh];');
fprintf(fileID, '%s\r\n', 'set e3   [expr $e2+1*$esh];');
fprintf(fileID, '%s\r\n', 'set e3n [expr $e2+((1.-
$Res)*$Fy+$esh*$b*$E)/($bd*$E)];');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial Steel02  20202 [expr $Fy*$Res]
[expr $E*$Res] [expr $b*0.000001] 20 0.925 0.15');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial Hysteretic 30303 $F1 $e1 $F2
$e2 $F3 $e3 -$F1 -$e1 -$F2 -$e2 -$F3n -$e3n 0.2 0.8 0.0 0.0');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial Parallel 40404 20202 30303 -
factors 1 1');


fprintf(fileID, '%s\r\n', 'set matTagC 40404');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# Bases');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    666   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    331   10e+8');
fprintf(fileID, '%s\r\n', 'uniaxialMaterial  Elastic    332   10e+8');
fprintf(fileID, '%s\r\n', '');
```

ANEXO J1: FUNCIÓN DE MATLAB PARA DEFINIR ELEMENTOS "ZERO LENGTH" EN MODELOS FIBER HINGE Y FIBER

```
function ZeroLength(NumS)

fileID = fopen(['ZeroLengthElem','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# PROPERTIES OF SPRINGS FOR THE PLASTIC
HINGES');
fprintf(fileID, '%s\r\n', '');

%COLUMN SPRINGS:

a=3;
pos_splices=zeros(1,NumS);
for i=1:NumS;

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;

end
num_sp=size(pos_splices);
num_splices=num_sp(1,2);
k=1;

for i=1:1


        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#ZERO LENGTH ELEMENTS FOR STORY: ',
i);
        fprintf(fileID, '%s\r\n', '# Spring ID: "3xya", where 3 = col
spring, x = Pier #, y = Story #, a = location in story');
        fprintf(fileID, '%s\r\n', '# "a" convention: 1 = bottom of story, 2
= top of story, 3=splice');
        fprintf(fileID, '%s\r\n', '');

           mat1=(['3','3','1']);
              mat2=(['3','3','2']);

         for j=1:4
            if j==1 || j==4
              ColumnTag1=(['3',num2str(j),num2str(i),'1']);

              node1 = ([num2str(j), num2str(i)]); %num2str = converts a
numeric array into a character array
              node2 = ([num2str(j), num2str(i),'7']);
              fprintf(fileID, '%18s %10s %6s %6s %6s %6s %6s %6s %12s\r\n',
'element zeroLength', ColumnTag1, node1, node2, '-mat ', '666 666', mat1,
'-dir ', '1 2 6');
              fprintf(fileID, '%s\r\n', '');
```

```
%               fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF',
node1, node2, '1', '2');
               fprintf(fileID, '%s\r\n', '');

            else
                ColumnTag1=(['3',num2str(j),num2str(i),'1']);

               node1 = ([num2str(j), num2str(i)]); %num2str = converts a
numeric array into a character array
               node2 = ([num2str(j), num2str(i),'7']);
               fprintf(fileID, '%18s %10s %6s %6s %6s %6s %6s %6s %12s\r\n',
'element zeroLength', ColumnTag1, node1, node2, '-mat ', '666 666', mat2,
'-dir ', '1 2 6');
               fprintf(fileID, '%s\r\n', '');
%               fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF',
node1, node2, '1', '2');
               fprintf(fileID, '%s\r\n', '');

            end

        end
end


% PANEL ZONE ELEMENTS:

for i=2:NumS+1
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', '#ZERO LENGTH FOR THE PANEL ZONE
FLOOR:',i);
            fprintf(fileID, '%s\r\n', '# eleID convention:  4xy00, 4 =
panel zone spring, x = Pier #, y = Floor #');

    for j=1:4

            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', ' #PIER: ',j);
            fprintf(fileID, '%s\r\n', '');

            PanelTag=(['4',num2str(j),num2str(i),'00']);
            node1 = ([num2str(j),num2str(i),'003']); %num2str = converts a
numeric array into a character array
            node2 = ([num2str(j),num2str(i),'004']);
            %fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag1, node1, node2,
10000000,29000, 1000000, '$PDeltaTransf;');
            fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', PanelTag, node1, node2, '-mat ', PanelTag, '-dir ',
'6');
            fprintf(fileID, '%s\r\n', '');


    end
end
fclose(fileID);
end
```

ANEXO J2: FUNCIÓN DE MATLAB PARA DEFINIR ELEMENTOS "ZERO LENGTH" EN MODELOS FIBER HINGE AND CPH Y FIBER AND CPH

```
function ZeroLength(NumS)

fileID = fopen(['ZeroLengthElem','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# PROPERTIES OF SPRINGS FOR THE PLASTIC
HINGES');
fprintf(fileID, '%s\r\n', '');

%COLUMN SPRINGS:

a=3;
pos_splices=zeros(1,NumS);
for i=1:NumS;

    if a>=NumS
        break;
    end
    pos_splices(a)=a;

    a=a+2;

end
num_sp=size(pos_splices);
num_splices=num_sp(1,2);
% k=1;

for i=1:1


        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#ZERO LENGTH ELEMENTS FOR STORY: ',
i);
        fprintf(fileID, '%s\r\n', '# Spring ID: "3xya", where 3 = col
spring, x = Pier #, y = Story #, a = location in story');
        fprintf(fileID, '%s\r\n', '# "a" convention: 1 = bottom of story, 2
= top of story, 3=splice');
        fprintf(fileID, '%s\r\n', '');

           mat1=(['3','3','1']);
              mat2=(['3','3','2']);

         for j=1:4
            if j==1 || j==4
              ColumnTag1=(['3',num2str(j),num2str(i),'1']);

              node1 = ([num2str(j), num2str(i)]); %num2str = converts a
numeric array into a character array
              node2 = ([num2str(j), num2str(i),'7']);
              fprintf(fileID, '%18s %10s %6s %6s %6s %6s %6s %6s %12s\r\n',
'element zeroLength', ColumnTag1, node1, node2, '-mat ', '666 666', mat1,
'-dir ', '1 2 6');
              fprintf(fileID, '%s\r\n', '');
```

```
%               fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF',
node1, node2, '1', '2');
               fprintf(fileID, '%s\r\n', '');

            else
                ColumnTag1=(['3',num2str(j),num2str(i),'1']);

               node1 = ([num2str(j), num2str(i)]); %num2str = converts a
numeric array into a character array
               node2 = ([num2str(j), num2str(i),'7']);
               fprintf(fileID, '%18s %10s %6s %6s %6s %6s %6s %6s %12s\r\n',
'element zeroLength', ColumnTag1, node1, node2, '-mat ', '666 666', mat2,
'-dir ', '1 2 6');
               fprintf(fileID, '%s\r\n', '');
%               fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF',
node1, node2, '1', '2');
               fprintf(fileID, '%s\r\n', '');

            end

        end
end
```

```
%BEAM ELEMENTS:

for i=2:NumS+1;

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#ZERO LENGTH ELEMENTS FOR FLOOR: ', i);
     fprintf(fileID, '%s\r\n', '# Spring ID: "4xya", where 3 = col spring,
x = Pier #, y = Story #, a = location in story');
     fprintf(fileID, '%s\r\n', '# "a" convention: 1 = left, 2 = right');
     fprintf(fileID, '%s\r\n', '');


     BeamRBSTag1=(['4','1',num2str(i),'2']);
     nodeRBS1 = (['1', num2str(i),'1']); %num2str = converts a numeric
array into a character array
     nodeRBS2 = (['1', num2str(i),'2']);
     fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', BeamRBSTag1, nodeRBS1, nodeRBS2, '-mat ',
BeamRBSTag1, '-dir ', '6');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodeRBS1,
nodeRBS2, '1', '2');
     fprintf(fileID, '%s\r\n', '');

     BeamRBSTag2=(['4','2',num2str(i),'1']);
     nodeRBS3 = (['2', num2str(i),'3']); %num2str = converts a numeric
array into a character array
     nodeRBS4 = (['2', num2str(i),'4']);
```

```
     fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', BeamRBSTag2, nodeRBS3, nodeRBS4, '-mat ',
BeamRBSTag2, '-dir ', '6');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodeRBS3,
nodeRBS4, '1', '2');
     fprintf(fileID, '%s\r\n', '');

     BeamRBSTag3=(['4','2',num2str(i),'2']);
     nodeRBS5 = (['2', num2str(i),'1']); %num2str = converts a numeric
array into a character array
     nodeRBS6 = (['2', num2str(i),'2']);
     fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', BeamRBSTag3, nodeRBS5, nodeRBS6, '-mat ',
BeamRBSTag3, '-dir ', '6');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodeRBS5,
nodeRBS6, '1', '2');
     fprintf(fileID, '%s\r\n', '');

     BeamRBSTag4=(['4','3',num2str(i),'1']);
     nodeRBS7 = (['3', num2str(i),'3']); %num2str = converts a numeric
array into a character array
     nodeRBS8 = (['3', num2str(i),'4']);
     fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', BeamRBSTag4, nodeRBS7, nodeRBS8, '-mat ',
BeamRBSTag4, '-dir ', '6');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodeRBS7,
nodeRBS8, '1', '2');
     fprintf(fileID, '%s\r\n', '');

     BeamRBSTag5=(['4','3',num2str(i),'2']);
     nodeRBS9 = (['3', num2str(i),'1']); %num2str = converts a numeric
array into a character array
     nodeRBS10 = (['3', num2str(i),'2']);
     fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', BeamRBSTag5, nodeRBS9, nodeRBS10, '-mat ',
BeamRBSTag5, '-dir ', '6');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodeRBS9,
nodeRBS10, '1', '2');
     fprintf(fileID, '%s\r\n', '');

     BeamRBSTag6=(['4','4',num2str(i),'1']);
     nodeRBS11= (['4', num2str(i),'3']); %num2str = converts a numeric
array into a character array
     nodeRBS12= (['4', num2str(i),'4']);
     fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', BeamRBSTag6, nodeRBS11, nodeRBS12, '-mat ',
BeamRBSTag6, '-dir ', '6');
     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', nodeRBS11,
nodeRBS12, '1', '2');
     fprintf(fileID, '%s\r\n', '');
```

```
end

% PANEL ZONE ELEMENTS:

for i=2:NumS+1
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', '#ZERO LENGTH FOR THE PANEL ZONE
FLOOR:',i);
            fprintf(fileID, '%s\r\n', '# eleID convention:  4xy00, 4 =
panel zone spring, x = Pier #, y = Floor #');

    for j=1:4

            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%s %u\r\n', ' #PIER: ',j);
            fprintf(fileID, '%s\r\n', '');

            PanelTag=(['4',num2str(j),num2str(i),'00']);
            node1 = ([num2str(j),num2str(i),'003']); %num2str = converts a
numeric array into a character array
            node2 = ([num2str(j),num2str(i),'004']);
            %fprintf(fileID, '%25s %6s %6s %6s %12.4e %12.4e %12.4e
%12s\r\n', 'element elasticBeamColumn', PanelElemTag1, node1, node2,
10000000,29000, 1000000, '$PDeltaTransf;');
            fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', PanelTag, node1, node2, '-mat ', PanelTag, '-dir ',
'6');
            fprintf(fileID, '%s\r\n', '');



    end
end
fclose(fileID);
end
```

ANEXO K: PROCESO PARA REALIZAR EL FIBRADO DE LAS SECCIONES

Tomado de (Martinez Pesantez & Pozo Ocampo, 2018)

```
##----------------------------------------------CREATE W SECTION-------------------------
-------------------##
# Formal arguments
# Tag - Section Tag # MatTag - Material Tag
# d - Depth # bf - Flange width
# tw - Web thickness # tf - Flange thickness
# k - Palm Ratio Web Coefficient # k1 - Palm Ratio Flange Coefficient
# SHEAR - YES OR NO; if includes Shear # Fy - Fluence Point un Axial Material
# G - Elastic Modulus of Shear # Alpha - Strain hardening for Shear Curve
##----------------------------------------------------------------------------------------
-----------------##
proc CreateWSection {Tag MatTag d bf tw tf k k1 {SHEAR NO} {vy 1} {G 1}
{Alpha 0.3}} {
##----------------------------------------------------------------------------------------
-----------------##
# SECTION EQUIVALENT
set pi 3.1416;
set h [expr $d-2*$tf];
set R [expr $k1 - $tw/(($k-$tf)/($k1-$tw/2.))];
set he [expr $tf];
set be [expr -(3*($R*$R*$R*$R)*(5*$pi - 16) - 4*($R*$R*$R)*(3*$pi - 10)*$h \
+ 12*($R*$R)*($pi*(1 - $tf*$d) - $h*$h) + $pi*($tf*$tf+3*$R*$R*$d*$d)) \
/(4*$tf*(3*$d*$d - 18*$d*$tf + 28*$tf*$tf))] ;
##----------------------------------------------------------------------------------------
-----------------##
# DEFINE SHEAR INCLUDE
if {$SHEAR==YES} {
set AditionalTag $Tag; set Tag [expr $Tag+100000];
}
##----------------------------------------------------------------------------------------
-----------------##
# DEFINE SECTION
section Fiber $Tag {
# patch rect $matTag $numSubdivY $numSubdivZ $YI $ZI \
$YJ $ZJ
# TOP FLANGE
patch rect $MatTag 5 2 [expr $d/2] [expr $bf/2]\
[expr $d/2-$tf] [expr -$bf/2];
# BTM FLANGE
patch rect $MatTag 5 2 [expr -$d/2+$tf] [expr $bf/2]\
[expr -$d/2] [expr -$bf/2];
# WEB
patch rect $MatTag 10 2 [expr $d/2-$tf] [expr $tw/2]\
[expr -$d/2+$tf] [expr -$tw/2];
```

```
# EQUIVALENT AREA
patch rect $MatTag 1 1 [expr $d/2-$tf] [expr ($tw/2)+$be]\
[expr $d/2-$tf-$he] [expr $tw/2];
patch rect $MatTag 1 1 [expr $d/2-$tf] [expr -$tw/2]\
[expr $d/2-$tf-$he] [expr (-$tw/2)-$be];
patch rect $MatTag 1 1 [expr -($d/2-$tf-$he)] [expr ($tw/2)+$be]\
[expr -($d/2-$tf)] [expr $tw/2];
patch rect $MatTag 1 1 [expr -($d/2-$tf-$he)] [expr -$tw/2]\
[expr -($d/2-$tf)] [expr (-$tw/2)-$be];
}

##-----------------------------------------------------------------------------------------------------------
-----------------##
# DEFINE SHEAR PROPERTIES
if {$SHEAR==YES} {
# DEFINE SHEAR CURVE
set Ke [expr 0.95*$G*$tw*$d]; set Kp [expr $Ke*$Alpha];
set y1 [expr $vy/(sqrt(3.)*$G)]; set V1 [expr 0.55*$vy*$d*$tw];
set y2 [expr 75.0*$y1*1]; set V2 [expr $V1+$Kp*($y2-$y1)];
set y3 [expr 100.0 *$y1*1]; set V3 [expr $V2+$Kp*($y3-$y2)];
# ADD SHEAR CURVE TO THE SECTION
uniaxialMaterial Hysteretic $Tag $V1 $y1 $V2 $y2 $V3 $y3 \
-$V1 -$y1 -$V2 -$y2 -$V3 -$y3 1 1 0.0 0.0 0.0
section Aggregator $AditionalTag $Tag Vy -section $Tag
puts "Section $AditionalTag \tinclude shear material with\t Vy: $V1\t G:\
$G\t Alpha:$Alpha"
}
}

#puts "Procedure CreateWSection \n "
#puts "CreateWSection SectionTag MatTag d bf tw tf k k1 <SHEAR vy G Alpha>"
```

ANEXO L1: CODIGO PARA EJECUTAR LOS ARCHIVOS TCL PRODUCIDOS POR MATLAB EN ESTRUCTURAS DE 2 PISOS

Este Código sirve también para la parte dinámica, para lo cual solo es necesario activar por medio de "#" en la parte que dice "Define análisis Type". Después es necesario desactivar los recorders, de igual manera solo con "#" (al agregarlo al principio de la línea, coloca en comentario la línea). De igual manera es necesario cambiar el numero de pisos, si el caso lo requiere.

```
# Element ID conventions:
#       1xy    = frame columns with RBS springs at both ends
#       2xy    = frame beams with RBS springs at both ends
#       6xy    = trusses linking frame and P-delta column
#       7xy    = P-delta columns
#       2xya   = frame beams between panel zone and RBS spring
#       3xya   = frame column rotational springs
#       4xya   = frame beam rotational springs
#       5xya   = P-delta column rotational springs
#       4xy00  = panel zone rotational springs
#       500xya = panel zone elements
#       where:
#               x = Pier or Bay #
#               y = Floor or Story #
#               a = an integer describing the location relative to beam-column joint (see description where elements and nodes are defined)

#############################################################################################
#       Set Up & Source Definition

#############################################################################################
        wipe all;                               # clear memory of past model definitions
        model BasicBuilder -ndm 2 -ndf 3;       # Define the model builder, ndm = #dimension, ndf = #dofs
        source DisplayModel2D.tcl;              # procedure for displaying a 2D perspective of model
        source DisplayPlane.tcl;                # procedure for displaying a plane in a model

#############################################################################################
#       Define Analysis Type
```

```
##########################################################################################
#########################

  # Define type of analysis:  "pushover" = pushover;  "dynamic" = dynamic
        #set analysisType "dynamic";
        set analysisType "pushover"



        if {$analysisType == "pushover"} {
                set dataDir Concentrated-PanelZone-Pushover-Output;      # name of output
folder
                file mkdir $dataDir;                                     # create output folder
        }



##########################################################################################
#########################
#       Define Building Geometry, Nodes, Masses, and Constraints

##########################################################################################
#########################
        set n 10;

        set NStories 2;                         # number of stories
        set NBays 3;                            # number of frame bays (excludes bay for P-delta
column)
        set WBay      [expr 20.0*12.0];                 # bay width in inches
        set HStory1   [expr 15.0*12.0];                 # 1st story height in inches
        set HStoryTyp [expr 13.0*12.0];                 # story height of other stories in inches
        set HBuilding [expr $HStory1 + ($NStories-1)*$HStoryTyp];       # height of building




# assign boundary condidtions
        source Nodes.tcl
        source Constraints.tcl
        # command:  fix nodeID dxFixity dyFixity rzFixity
        # fixity values: 1 = constrained; 0 = unconstrained
        # fix the base of the building; pin P-delta column at base
        fix 11 1 1 1;
        fix 21 1 1 1;
        fix 31 1 1 1;
        fix 41 1 1 1;
        fix 51 1 1 0;    # P-delta column is pinned
```

```
set transfTag 1;
geomTransf PDelta  $transfTag;       # PDelta transformation



        source Material.tcl

        source CreateWSection.tcl
        source Sections.tcl
        source Elements.tcl
        region 4 -ele  111      121     131     141     112     122     132     142
        source ZeroLengthElem.tcl
        source Masses.tcl




#DisplayModel2D NodeNumbers



##########################################################################################
#
#           Gravity Loads & Gravity Analysis                                        #
##########################################################################################
#

# apply gravity loads
        #command: pattern PatternType $PatternID TimeSeriesType
        pattern Plain 101 Constant {

                # point loads on leaning column nodes
                # command: load node Fx Fy Mz
                set P_PD2 [expr -568.00];     # Floor 2

                set P_PD3 [expr -511.20];     # Floor 3

                load 52 0.0 $P_PD2 0.0;                 # Floor 2
                load 53 0.0 $P_PD3 0.0;                 # Floor 3



                # point loads on frame column nodes
```

```
        set P_F21 [expr -70.21];        # load on each frame node in Floor 2 (exterior)
        set P_F22 [expr -46.14];        # load on each frame node in Floor 2 (interior)
        set P_F31 [expr -61.77];        # load on each frame node in Floor 3 (exterior)
        set P_F32 [expr -41.18];        # load on each frame node in Floor 3 (interior)



        # Floor 2 loads
        load 127 0.0 $P_F21 0.0;
        load 227 0.0 $P_F22 0.0;
        load 327 0.0 $P_F22 0.0;
        load 427 0.0 $P_F21 0.0;

        # Floor 3 loads
        load 137 0.0 $P_F31 0.0;
        load 237 0.0 $P_F32 0.0;
        load 337 0.0 $P_F32 0.0;
        load 437 0.0 $P_F31 0.0;




    }

# Gravity-analysis: load-controlled static analysis
        set Tol 1.0e-6;                 # convergence tolerance for test
        constraints Plain;              # how it handles boundary conditions
        numberer RCM;                           # renumber dof's to minimize band-width
(optimization)
        system BandGeneral;             # how to store and solve the system of equations
in the analysis (large model: try UmfPack)
        test NormDispIncr $Tol 6;       # determine if convergence has been achieved at
the end of an iteration step
        algorithm Newton;               # use Newton's solution algorithm: updates
tangent stiffness at every iteration
        set NstepGravity 1;             # apply gravity in 10 steps
        set DGravity [expr 1.0/$NstepGravity];  # load increment
        integrator LoadControl $DGravity;  # determine the next time step for an analysis
        analysis Static;                # define type of analysis: static or transient
        analyze $NstepGravity;                  # apply gravity

        # maintain constant gravity loads and reset time to zero
        loadConst -time 0.0
        puts "Model Built"
```

```
##########################################################################
#
#                    Eigenvalue Analysis
##########################################################################
#

        set pi [expr 2.0*asin(1.0)];                    # Definition of pi
        set nEigenI 1;                                  # mode i = 1
        set nEigenJ 2;                                  # mode j = 2


        set lambdaN [eigen  -genBandArpack  [expr $nEigenJ]];                   # eigenvalue
analysis for nEigenQ modes
        set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i = 1
        set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j = 2


        set w1 [expr pow($lambdaI,0.5)];                # w1 (1st mode circular frequency)
        set w2 [expr pow($lambdaJ,0.5)];                # w2 (2nd mode circular
frequency)



        set T1 [expr 2.0*$pi/$w1];                      # 1st mode period of the structure
        set T2 [expr 2.0*$pi/$w2];                      # 2nd mode period of the structure


        puts "T1 = $T1 s";                              # display the first mode period in
the command window
        puts "T2 = $T2 s";                              # display the second mode period
in the command window


##########################################################################
############
#                                                     #
#                    Analysis Section                         #
#                                                     #
##########################################################################
############

##########################################################################
#
#          Pushover Analysis                                  #
##########################################################################
#

recorder Element -file $dataDir/Fcol_story_forces.out -region 4 force
```

```
recorder Node -file $dataDir/Disp_Roof.out -time -node 13005 -dof 1 disp;
#recorder Node -file $dataDir/Rot.out -time -node 11 21 31 41 51 -dof 1 2 3 disp;
#recorder Node -file $dataDir/Vbase.out -time -node 11 21 31 41  -dof 1 2 3 reaction;
#recorder Element -file $dataDir/MRFcol-Mom-1.out -time -ele 3111 3211 3311 3411 force;
#recorder Element -file $dataDir/MRFcol-Rot-1.out -time -ele 3111 3211 3311 3411
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-2.out -time -ele 3112 3212 3312 3412 force;
#recorder Element -file $dataDir/MRFcol-Rot-2.out -time -ele 3112 3212 3312 3412
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-3.out -time -ele 3121 3221 3321 3421 force;
#recorder Element -file $dataDir/MRFcol-Rot-3.out -time -ele 3121 3221 3321 3421
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-1.out -time -ele 4122 4221 force;
#recorder Element -file $dataDir/MRFbeam-Rot-1.out -time -ele 4122 4221 deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-2.out -time -ele 4132 4231 force;
#recorder Element -file $dataDir/MRFbeam-Rot-2.out -time -ele 4132 4231 deformation;

DisplayModel2D DeformedShape 1 300 200 1080 720

if {$analysisType == "pushover"} {
        puts "Running Pushover..."
# assign lateral loads and create load pattern:  use ASCE 7-10 distribution
        set lat2 0.302828065; # force on each beam-column joint in Floor 2
set lat3 0.697171935; # force on each beam-column joint in Floor 3




        pattern Plain 200 Linear {
                                        load 12005 $lat2 0.0 0.0;
                                        #load 2205 $lat2 0.0 0.0;
                                        #load 3205 $lat2 0.0 0.0;
                                        #load 4205 $lat2 0.0 0.0;
                                        load 13005 $lat3 0.0 0.0;
                                        #load 2305 $lat3 0.0 0.0;
                                        #load 3305 $lat3 0.0 0.0;
                                        #load 4305 $lat3 0.0 0.0;


        }

# display deformed shape:
        set ViewScale 5;
        #DisplayModel2D DeformedShape $ViewScale ;    # display deformed shape, the
scaling factor needs to be adjusted for each model

# displacement parameters
```

```
	set IDctrlNode 13005;				# node where disp is read for disp
control
	set IDctrlDOF 1;				# degree of freedom read for disp control
(1 = x displacement)
	set Dmax [expr 0.90*$HBuilding];		# maximum displacement of pushover:
4% roof drift
	set Dincr [expr 0.1];				# displacement increment

# analysis commands
	constraints Plain;				# how it handles boundary conditions
	numberer Plain;					# renumber dof's to minimize
band-width (optimization)
	system UmfPack;					# how to store and solve the system of
equations in the analysis (large model: try UmfPack)
	set Tol 1.e-4;					# Convergence Test: tolerance
	set maxNumIter 100;				# Convergence Test: maximum number of
iterations that will be performed before "failure to converge" is returned
	set printFlag 0;
	set TestType EnergyIncr;

	#test EnergyIncr  1.0e-5 400;			# type of convergence criteria with
tolerance, max iterations
	#test $TestType $Tol $maxNumIter $printFlag;
	test NormUnbalance 0.01 100;

	set algorithmType Newton;
	algorithm $algorithmType;


	#algorithm Newton;				# use Newton's solution algorithm:
updates tangent stiffness at every iteration
	integrator DisplacementControl  $IDctrlNode   $IDctrlDOF $Dincr;	# use
displacement-controlled analysis
	analysis Static;				# define type of analysis: static for
pushover
	set Nsteps [expr int($Dmax/$Dincr)];# number of pushover analysis steps

	for {set i 1} {$i <= $Nsteps} {incr i 1} {
		set ok [analyze 1];			# this will return zero if no convergence
problems were encountered
		puts "PO step $i out of $Nsteps: $ok";
	    if {$ok != 0} {
			source PO_convergence_loop.tcl;
		};
	};
```

```
	#set ok [analyze $Nsteps];			# this will return zero if no convergence
problems were encountered
	puts "Pushover complete";			# display this message in the command
window
}

##########################################################################
#
#   Time History/Dynamic Analysis                                        #
##########################################################################
#

if {$analysisType == "dynamic"} {
	puts "Running dynamic analysis..."

	source dynamic.tcl

}

#wipe all;
```

ANEXO L2: CODIGO PARA EJECUTAR LOS ARCHIVOS TCL PRODUCIDOS POR MATLAB EN ESTRUCTURAS DE 4 PISOS

Este Código sirve también para la parte dinámica, para lo cual solo es necesario activar por medio de "#" en la parte que dice "Define análisis Type". Después es necesario desactivar los recorders, de igual manera solo con "#" (al agregarlo al principio de la línea, coloca en comentario la línea)

```
# Element ID conventions:
#       1xy    = frame columns with RBS springs at both ends
#       2xy    = frame beams with RBS springs at both ends
#       6xy    = trusses linking frame and P-delta column
#       7xy    = P-delta columns
#       2xya   = frame beams between panel zone and RBS spring
#       3xya   = frame column rotational springs
#       4xya   = frame beam rotational springs
#       5xya   = P-delta column rotational springs
#       4xy00  = panel zone rotational springs
#       500xya = panel zone elements
#       where:
#               x = Pier or Bay #
#               y = Floor or Story #
#               a = an integer describing the location relative to beam-column joint (see description where elements and nodes are defined)

#########################################################################################
#       Set Up & Source Definition

#########################################################################################
        wipe all;                               # clear memory of past model definitions
        model BasicBuilder -ndm 2 -ndf 3;       # Define the model builder, ndm = #dimension, ndf = #dofs
        source DisplayModel2D.tcl;              # procedure for displaying a 2D perspective of model
        source DisplayPlane.tcl;                # procedure for displaying a plane in a model

#########################################################################################
#       Define Analysis Type

#########################################################################################
```

```
 # Define type of analysis:  "pushover" = pushover;  "dynamic" = dynamic
        #set analysisType "dynamic";
        set analysisType "pushover"



        if {$analysisType == "pushover"} {
                set dataDir Concentrated-PanelZone-Pushover-Output;      # name of
output folder
                file mkdir $dataDir;                                     # create output
folder
        }



##########################################################################
################################
#         Define Building Geometry, Nodes, Masses, and Constraints

##########################################################################
################################
        set n 10;

        set NStories 4;                         # number of stories
        set NBays 3;                            # number of frame bays (excludes bay for
P-delta column)
        set WBay      [expr 20.0*12.0];             # bay width in inches
        set HStory1   [expr 15.0*12.0];             # 1st story height in inches
        set HStoryTyp [expr 13.0*12.0];             # story height of other stories in
inches
        set HBuilding [expr $HStory1 + ($NStories-1)*$HStoryTyp];       # height of
building




# assign boundary condidtions
        source Nodes.tcl
        source Constraints.tcl
        # command:  fix nodeID dxFixity dyFixity rzFixity
        # fixity values: 1 = constrained; 0 = unconstrained
        # fix the base of the building; pin P-delta column at base
        fix 11 1 1 1;
        fix 21 1 1 1;
        fix 31 1 1 1;
        fix 41 1 1 1;
```

```
        fix 51 1 1 0;    # P-delta column is pinned


set transfTag 1;
geomTransf PDelta  $transfTag;       # PDelta transformation



        source Material.tcl

        source CreateWSection.tcl
        source Sections.tcl
        source Elements.tcl
        region 4 -ele  111      121     131     141     112     122     132     142
        11391  11392  12391  12392  13391  13392  14391  14392  114     124     134
        144
        source ZeroLengthElem.tcl
        source Masses.tcl

#source Elements.tcl



#DisplayModel2D NodeNumbers
#source Material.tcl
#source ZeroLengthElem.tcl



#########################################################################
#######
#           Gravity Loads & Gravity Analysis                                    #
#########################################################################
#######

# apply gravity loads
        #command: pattern PatternType $PatternID TimeSeriesType
        pattern Plain 101 Constant {

                # point loads on leaning column nodes
                # command: load node Fx Fy Mz
                set P_PD2 [expr -568.00];     # Floor 2
                set P_PD3 [expr -565.20];     # Floor 3
                set P_PD5 [expr -511.20];     # Floor 5

                load 52 0.0 $P_PD2 0.0;               # Floor 2
                load 53 0.0 $P_PD3 0.0;               # Floor 3
                load 54 0.0 $P_PD3 0.0;               # Floor 4
```

```
load 55 0.0 $P_PD5 0.0;                # Floor 5


# point loads on frame column nodes

set P_F21 [expr -70.21];       # load on each frame node in Floor 2
(exterior)
set P_F22 [expr -46.14];       # load on each frame node in Floor 2
(interior)
set P_F31 [expr -70.21];       # load on each frame node in Floor 3
(exterior)
set P_F32 [expr -46.14];       # load on each frame node in Floor 3
(interior)
set P_F41 [expr -70.21];       # load on each frame node in Floor 4
(exterior)
set P_F42 [expr -46.14];       # load on each frame node in Floor 4
(interior)
set P_F51 [expr -61.77];       # load on each frame node in Floor 5
(exterior)
set P_F52 [expr -41.18];       # load on each frame node in Floor 5
(interior)


# Floor 2 loads
load 127 0.0 $P_F21 0.0;
load 227 0.0 $P_F22 0.0;
load 327 0.0 $P_F22 0.0;
load 427 0.0 $P_F21 0.0;

# Floor 3 loads
load 137 0.0 $P_F31 0.0;
load 237 0.0 $P_F32 0.0;
load 337 0.0 $P_F32 0.0;
load 437 0.0 $P_F31 0.0;

# Floor 4 loads
load 147 0.0 $P_F41 0.0;
load 247 0.0 $P_F42 0.0;
load 347 0.0 $P_F42 0.0;
load 447 0.0 $P_F41 0.0;

# Floor 5 loads
load 157 0.0 $P_F51 0.0;
load 257 0.0 $P_F52 0.0;
load 357 0.0 $P_F52 0.0;
```

```
                load 457 0.0 $P_F51 0.0;




        }

# Gravity-analysis: load-controlled static analysis
        set Tol 1.0e-6;                         # convergence tolerance for test
        constraints Plain;                      # how it handles boundary conditions
        numberer RCM;                                   # renumber dof's to minimize
band-width (optimization)
        system BandGeneral;                     # how to store and solve the system of
equations in the analysis (large model: try UmfPack)
        test NormDispIncr $Tol 6;               # determine if convergence has been
achieved at the end of an iteration step
        algorithm Newton;                       # use Newton's solution algorithm:
updates tangent stiffness at every iteration
        set NstepGravity 1;                     # apply gravity in 10 steps
        set DGravity [expr 1.0/$NstepGravity];      # load increment
        integrator LoadControl $DGravity;   # determine the next time step for an
analysis
        analysis Static;                        # define type of analysis: static or
transient
        analyze $NstepGravity;                          # apply gravity

        # maintain constant gravity loads and reset time to zero
        loadConst -time 0.0
        puts "Model Built"


#########################################################################
#######
#                   Eigenvalue Analysis
#########################################################################
#######

        set pi [expr 2.0*asin(1.0)];                            # Definition of pi
        set nEigenI 1;                                          # mode i = 1
        set nEigenJ 2;                                          # mode j = 2
                                                                        # mode k = 3
        set nEigenL 4;                                          # mode l = 4



        set lambdaN [eigen  -genBandArpack  [expr $nEigenL]];                   #
eigenvalue analysis for nEigenQ modes
```

```
	set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i = 1
	set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j = 2
	set lambdaK [lindex $lambdaN [expr $nEigenK-1]];	# eigenvalue mode
k = 3
	set lambdaL [lindex $lambdaN [expr $nEigenL-1]];	# eigenvalue mode l
= 4


	set w1 [expr pow($lambdaI,0.5)];			# w1 (1st mode circular
frequency)
	set w2 [expr pow($lambdaJ,0.5)];			# w2 (2nd mode circular
frequency)
	set w3 [expr pow($lambdaK,0.5)];			# w3 (3st mode circular
frequency)
	set w4 [expr pow($lambdaL,0.5)];			# w4 (4nd mode circular
frequency)



	set T1 [expr 2.0*$pi/$w1];				# 1st mode period of the
structure
	set T2 [expr 2.0*$pi/$w2];				# 2nd mode period of the
structure
	set T3 [expr 2.0*$pi/$w3];				# 3st mode period of the
structure
	set T4 [expr 2.0*$pi/$w4];				# 4nd mode period of the
structure


	puts "T1 = $T1 s";					# display the first mode
period in the command window
	puts "T2 = $T2 s";					# display the second mode
period in the command window
	puts "T3 = $T3 s";					# display the third mode
period in the command window
	puts "T4 = $T4 s";					# display the fourth mode
period in the command window


##########################################################################
##################
#                                                                        #
#                    Analysis Section                                    #
#                                                                        #
##########################################################################
##################
```

```
#########################################################################
#######
#            Pushover Analysis                                          #
#########################################################################
#######

recorder Element -file $dataDir/Fcol_story_forces.out -region 4 force
recorder Node -file $dataDir/Disp_Roof.out -time -node 15005 -dof 1 disp;
#recorder Node -file $dataDir/Rot.out -time -node 11 21 31 41 51 -dof 1 2 3 disp;
#recorder Node -file $dataDir/Vbase.out -time -node 11 21 31 41  -dof 1 2 3 reaction;
#recorder Element -file $dataDir/MRFcol-Mom-1.out -time -ele 3111 3211 3311 3411
force;
#recorder Element -file $dataDir/MRFcol-Rot-1.out -time -ele 3111 3211 3311 3411
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-2.out -time -ele 3112 3212 3312 3412
force;
#recorder Element -file $dataDir/MRFcol-Rot-2.out -time -ele 3112 3212 3312 3412
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-3.out -time -ele 3121 3221 3321 3421
force;
#recorder Element -file $dataDir/MRFcol-Rot-3.out -time -ele 3121 3221 3321 3421
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-1.out -time -ele 4122 4221 force;
#recorder Element -file $dataDir/MRFbeam-Rot-1.out -time -ele 4122 4221
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-2.out -time -ele 4132 4231 force;
#recorder Element -file $dataDir/MRFbeam-Rot-2.out -time -ele 4132 4231
deformation;

DisplayModel2D DeformedShape 1 300 200 1080 720

if {$analysisType == "pushover"} {
        puts "Running Pushover..."
# assign lateral loads and create load pattern:  use ASCE 7-10 distribution
        set lat2 0.095499716; # force on each beam-column joint in Floor 2
set lat3 0.207725251; # force on each beam-column joint in Floor 3
set lat4 0.312070995; # force on each beam-column joint in Floor 4
set lat5 0.384704039; # force on each beam-column joint in Floor 5



        pattern Plain 200 Linear {
                                        load 12005 $lat2 0.0 0.0;
                                        #load 2205 $lat2 0.0 0.0;
                                        #load 3205 $lat2 0.0 0.0;
                                        #load 4205 $lat2 0.0 0.0;
                                        load 13005 $lat3 0.0 0.0;
                                        #load 2305 $lat3 0.0 0.0;
```

```
                                        #load 3305 $lat3 0.0 0.0;
                                        #load 4305 $lat3 0.0 0.0;
                                        load 14005 $lat4 0.0 0.0;
                                        #load 2405 $lat4 0.0 0.0;
                                        #load 3405 $lat4 0.0 0.0;
                                        #load 4405 $lat4 0.0 0.0;
                                        load 15005 $lat5 0.0 0.0;
                                        #load 2505 $lat5 0.0 0.0;
                                        #load 3505 $lat5 0.0 0.0;
                                        #load 4505 $lat5 0.0 0.0;


        }

# display deformed shape:
        set ViewScale 5;
        #DisplayModel2D DeformedShape $ViewScale ;   # display deformed shape,
the scaling factor needs to be adjusted for each model

# displacement parameters
        set IDctrlNode 15005;                           # node where disp is read
for disp control
        set IDctrlDOF 1;                        # degree of freedom read for disp
control (1 = x displacement)
        set Dmax [expr 0.9*$HBuilding];         # maximum displacement of
pushover: 4% roof drift
        set Dincr [expr 0.1];                   # displacement increment

# analysis commands
        constraints Plain;                      # how it handles boundary
conditions
        numberer Plain;                                 # renumber dof's to
minimize band-width (optimization)
        system UmfPack;                         # how to store and solve the
system of equations in the analysis (large model: try UmfPack)
        set Tol 1.e-4;                          # Convergence Test: tolerance
        set maxNumIter 100;                     # Convergence Test: maximum
number of iterations that will be performed before "failure to converge" is returned
        set printFlag 0;
        set TestType EnergyIncr;

        #test EnergyIncr  1.0e-5 400;           # type of convergence criteria with
tolerance, max iterations
        #test $TestType $Tol $maxNumIter $printFlag;
        test NormUnbalance 0.01 100;

        set algorithmType Newton;
        algorithm $algorithmType;
```

```
	#algorithm Newton;				# use Newton's solution algorithm: updates tangent stiffness at every iteration
	integrator DisplacementControl  $IDctrlNode   $IDctrlDOF $Dincr;	# use displacement-controlled analysis
	analysis Static;				# define type of analysis: static for pushover
	set Nsteps [expr int($Dmax/$Dincr)];# number of pushover analysis steps

	for {set i 1} {$i <= $Nsteps} {incr i 1} {
		set ok [analyze 1];			# this will return zero if no convergence problems were encountered
		puts "PO step $i out of $Nsteps: $ok";
	    if {$ok != 0} {
			source PO_convergence_loop.tcl;
		};
	};



	#set ok [analyze $Nsteps];			# this will return zero if no convergence problems were encountered
	puts "Pushover complete";			# display this message in the command window
}

##########################################################################
#######
#   Time History/Dynamic Analysis                                        #
##########################################################################
#######

if {$analysisType == "dynamic"} {
	puts "Running dynamic analysis..."

	source dynamic.tcl

}

#wipe all;
```

ANEXO L3: CODIGO PARA EJECUTAR LOS ARCHIVOS TCL PRODUCIDOS POR MATLAB EN ESTRUCTURAS DE 8 PISOS

Este Código sirve también para la parte dinámica, para lo cual solo es necesario activar por medio de "#" en la parte que dice "Define análisis Type". Después es necesario desactivar los recorders, de igual manera solo con "#" (al agregarlo al principio de la línea, coloca en comentario la línea)

```
# Element ID conventions:
#       1xy   = frame columns with RBS springs at both ends
#       2xy   = frame beams with RBS springs at both ends
#       6xy   = trusses linking frame and P-delta column
#       7xy   = P-delta columns
#       2xya  = frame beams between panel zone and RBS spring
#       3xya  = frame column rotational springs
#       4xya  = frame beam rotational springs
#       5xya  = P-delta column rotational springs
#       4xy00 = panel zone rotational springs
#       500xya = panel zone elements
#       where:
#               x = Pier or Bay #
#               y = Floor or Story #
#               a = an integer describing the location relative to beam-column joint (see description where elements and nodes are defined)

#########################################################################################
#       Set Up & Source Definition

#########################################################################################
        wipe all;                               # clear memory of past model definitions
        model BasicBuilder -ndm 2 -ndf 3;       # Define the model builder, ndm = #dimension, ndf = #dofs
        source DisplayModel2D.tcl;              # procedure for displaying a 2D perspective of model
        source DisplayPlane.tcl;                # procedure for displaying a plane in a model

#########################################################################################
#       Define Analysis Type
```

```
#########################################################################
##############################

 # Define type of analysis:  "pushover" = pushover;  "dynamic" = dynamic
        #set analysisType "dynamic";
        set analysisType "pushover"



        if {$analysisType == "pushover"} {
                set dataDir Concentrated-PanelZone-Pushover-Output;      # name of
output folder
                file mkdir $dataDir;                                     # create output
folder
        }



#########################################################################
##############################
#       Define Building Geometry, Nodes, Masses, and Constraints

#########################################################################
##############################
        set n 10;

        set NStories 8;                 # number of stories
        set NBays 3;                    # number of frame bays (excludes bay for
P-delta column)
        set WBay     [expr 20.0*12.0];          # bay width in inches
        set HStory1   [expr 15.0*12.0];         # 1st story height in inches
        set HStoryTyp [expr 13.0*12.0];         # story height of other stories in
inches
        set HBuilding [expr $HStory1 + ($NStories-1)*$HStoryTyp];       # height of
building




# assign boundary condidtions
        source Nodes.tcl
        source Constraints.tcl
        # command:  fix nodeID dxFixity dyFixity rzFixity
        # fixity values: 1 = constrained; 0 = unconstrained
        # fix the base of the building; pin P-delta column at base
        fix 11 1 1 1;
        fix 21 1 1 1;
```

```
        fix 31 1 1 1;
        fix 41 1 1 1;
        fix 51 1 1 0;    # P-delta column is pinned


set transfTag 1;
geomTransf PDelta  $transfTag;       # PDelta transformation elastic elements



        source Material.tcl

        source CreateWSection.tcl
        source Sections.tcl
        source Elements.tcl
        region 4 -ele  111      121     131     141     112     122     132     142
        11391  11392  12391  12392  13391  13392  14391  14392  114     124     134
        144     11591  11592  12591  12592  13591  13592  14591  14592  116     126
        136     146     11791  11792  12791  12792  13791  13792  14791  14792  118
        128     138     148

        source ZeroLengthElem.tcl
        source Masses.tcl




#DisplayModel2D NodeNumbers



########################################################################
#######
#           Gravity Loads & Gravity Analysis                                    #
########################################################################
#######

# apply gravity loads
        #command: pattern PatternType $PatternID TimeSeriesType
        pattern Plain 101 Constant {

                # point loads on leaning column nodes
                # command: load node Fx Fy Mz
                set P_PD2 [expr -568.00];     # Floor 2
                set P_PD3 [expr -565.20];     # Floor 3
                set P_PD9 [expr -511.20];     # Floor 9
```

```
load 52 0.0 $P_PD2 0.0;          # Floor 2
load 53 0.0 $P_PD3 0.0;          # Floor 3
load 54 0.0 $P_PD3 0.0;          # Floor 4
load 55 0.0 $P_PD3 0.0;          # Floor 5
load 56 0.0 $P_PD3 0.0;          # Floor 6
load 57 0.0 $P_PD3 0.0;          # Floor 7
load 58 0.0 $P_PD3 0.0;          # Floor 8
load 59 0.0 $P_PD3 0.0;          # Floor 9

# point loads on frame column nodes

set P_F21 [expr -70.21];      # load on each frame node in Floor 2 (exterior)
set P_F22 [expr -46.14];      # load on each frame node in Floor 2 (interior)
set P_F31 [expr -70.21];      # load on each frame node in Floor 3 (exterior)
set P_F32 [expr -46.14];      # load on each frame node in Floor 3 (interior)
set P_F41 [expr -70.21];      # load on each frame node in Floor 4 (exterior)
set P_F42 [expr -46.14];      # load on each frame node in Floor 4 (interior)
set P_F51 [expr -70.21];      # load on each frame node in Floor 5 (exterior)
set P_F52 [expr -46.14];      # load on each frame node in Floor 5 (interior)
set P_F61 [expr -70.21];      # load on each frame node in Floor 6 (exterior)
set P_F62 [expr -46.14];      # load on each frame node in Floor 6 (interior)
set P_F71 [expr -70.21];      # load on each frame node in Floor 7 (exterior)
set P_F72 [expr -46.14];      # load on each frame node in Floor 7 (interior)
set P_F81 [expr -70.21];      # load on each frame node in Floor 8 (exterior)
set P_F82 [expr -46.14];      # load on each frame node in Floor 8 (interior)
set P_F91 [expr -61.77];      # load on each frame node in Floor 9 (exterior)
set P_F92 [expr -41.18];      # load on each frame node in Floor 9 (interior)
```

```
# Floor 2 loads
load 127 0.0 $P_F21 0.0;
load 227 0.0 $P_F22 0.0;
load 327 0.0 $P_F22 0.0;
load 427 0.0 $P_F21 0.0;

# Floor 3 loads
load 137 0.0 $P_F31 0.0;
load 237 0.0 $P_F32 0.0;
load 337 0.0 $P_F32 0.0;
load 437 0.0 $P_F31 0.0;

# Floor 4 loads
load 147 0.0 $P_F41 0.0;
load 247 0.0 $P_F42 0.0;
load 347 0.0 $P_F42 0.0;
load 447 0.0 $P_F41 0.0;

# Floor 5 loads
load 157 0.0 $P_F51 0.0;
load 257 0.0 $P_F52 0.0;
load 357 0.0 $P_F52 0.0;
load 457 0.0 $P_F51 0.0;

# Floor 6 loads
load 167 0.0 $P_F61 0.0;
load 267 0.0 $P_F62 0.0;
load 367 0.0 $P_F62 0.0;
load 467 0.0 $P_F61 0.0;

# Floor 7 loads
load 177 0.0 $P_F71 0.0;
load 277 0.0 $P_F72 0.0;
load 377 0.0 $P_F72 0.0;
load 477 0.0 $P_F71 0.0;

# Floor 8 loads
load 187 0.0 $P_F81 0.0;
load 287 0.0 $P_F82 0.0;
load 387 0.0 $P_F82 0.0;
load 487 0.0 $P_F81 0.0;

# Floor 9 loads
load 197 0.0 $P_F91 0.0;
load 297 0.0 $P_F92 0.0;
load 397 0.0 $P_F92 0.0;
load 497 0.0 $P_F91 0.0;
```

```
	}

# Gravity-analysis: load-controlled static analysis
	set Tol 1.0e-6;				# convergence tolerance for test
	constraints Plain;			# how it handles boundary conditions
	numberer RCM;					# renumber dof's to minimize
band-width (optimization)
	system BandGeneral;			# how to store and solve the system of
equations in the analysis (large model: try UmfPack)
	test NormDispIncr $Tol 6;		# determine if convergence has been
achieved at the end of an iteration step
	algorithm Newton;			# use Newton's solution algorithm:
updates tangent stiffness at every iteration
	set NstepGravity 1;			# apply gravity in 10 steps
	set DGravity [expr 1.0/$NstepGravity];	# load increment
	integrator LoadControl $DGravity;	# determine the next time step for an
analysis
	analysis Static;			# define type of analysis: static or
transient
	analyze $NstepGravity;			# apply gravity

	# maintain constant gravity loads and reset time to zero
	loadConst -time 0.0
	puts "Model Built"


##########################################################################
#######
#			Eigenvalue Analysis
##########################################################################
#######

	set pi [expr 2.0*asin(1.0)];				# Definition of pi
	set nEigenI 1;					# mode i = 1
	set nEigenJ 2;					# mode j = 2
	set nEigenK 3;						# mode k = 3
	set nEigenL 4;					# mode l = 4
	set nEigenM 5;						# mode m = 5
	set nEigenO 6;						# mode o = 6
	set nEigenP 7;					# mode p = 7
	set nEigenQ 8;						# mode q = 8

	set lambdaN [eigen  -genBandArpack  [expr $nEigenQ]];			#
eigenvalue analysis for nEigenQ modes
	set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i = 1
```

```
        set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j = 2
        set lambdaK [lindex $lambdaN [expr $nEigenK-1]];        # eigenvalue mode
k = 3
        set lambdaL [lindex $lambdaN [expr $nEigenL-1]];        # eigenvalue mode l
= 4
        set lambdaM [lindex $lambdaN [expr $nEigenM-1]];        # eigenvalue mode
m = 5
        set lambdaO [lindex $lambdaN [expr $nEigenO-1]];        # eigenvalue mode
o = 6
        set lambdaP [lindex $lambdaN [expr $nEigenP-1]];# eigenvalue mode p = 7
        set lambdaQ [lindex $lambdaN [expr $nEigenQ-1]];        # eigenvalue mode
q = 8

        set w1 [expr pow($lambdaI,0.5)];                # w1 (1st mode circular
frequency)
        set w2 [expr pow($lambdaJ,0.5)];                # w2 (2nd mode circular
frequency)
        set w3 [expr pow($lambdaK,0.5)];                # w3 (3st mode circular
frequency)
        set w4 [expr pow($lambdaL,0.5)];                # w4 (4nd mode circular
frequency)
        set w5 [expr pow($lambdaM,0.5)];                # w5 (5st mode circular
frequency)
        set w6 [expr pow($lambdaO,0.5)];                # w6 (6nd mode circular
frequency)
        set w7 [expr pow($lambdaP,0.5)];                # w7 (7st mode circular
frequency)
        set w8 [expr pow($lambdaQ,0.5)];                # w8 (8nd mode circular
frequency)


        set T1 [expr 2.0*$pi/$w1];                      # 1st mode period of the
structure
        set T2 [expr 2.0*$pi/$w2];                      # 2nd mode period of the
structure
        set T3 [expr 2.0*$pi/$w3];                      # 3st mode period of the
structure
        set T4 [expr 2.0*$pi/$w4];                      # 4nd mode period of the
structure
        set T5 [expr 2.0*$pi/$w5];                      # 5st mode period of the
structure
        set T6 [expr 2.0*$pi/$w6];                      # 6nd mode period of the
structure
        set T7 [expr 2.0*$pi/$w7];                      # 7st mode period of the
structure
        set T8 [expr 2.0*$pi/$w8];                      # 8nd mode period of the
structure
```

```
	puts "T1 = $T1 s";				# display the first mode
period in the command window
	puts "T2 = $T2 s";				# display the second mode
period in the command window
	puts "T3 = $T3 s";				# display the third mode
period in the command window
	puts "T4 = $T4 s";				# display the fourth mode
period in the command window
	puts "T5 = $T5 s";				# display the fifth mode
period in the command window
	puts "T6 = $T6 s";				# display the sixth mode
period in the command window
	puts "T7 = $T7 s";				# display the seventh mode
period in the command window
	puts "T8 = $T8 s";				# display the eight mode
period in the command window


##########################################################################
###################
#                                                       #
#                   Analysis Section                                #
#                                                       #
##########################################################################
###################

##########################################################################
#######
#          Pushover Analysis                                #
##########################################################################
#######

recorder Element -file $dataDir/Fcol_story_forces.out -region 4 force
recorder Node -file $dataDir/Disp_Roof.out -time -node 19005 -dof 1 disp;


#recorder Node -file $dataDir/Rot.out -time -node 11 21 31 41 51 -dof 1 2 3 disp;
#recorder Node -file $dataDir/Vbase.out -time -node 11 21 31 41  -dof 1 2 3 reaction;
#recorder Element -file $dataDir/MRFcol-Mom-1.out -time -ele 3111 3211 3311 3411
force;
#recorder Element -file $dataDir/MRFcol-Rot-1.out -time -ele 3111 3211 3311 3411
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-2.out -time -ele 3112 3212 3312 3412
force;
#recorder Element -file $dataDir/MRFcol-Rot-2.out -time -ele 3112 3212 3312 3412
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-3.out -time -ele 3121 3221 3321 3421
force;
```

```
#recorder Element -file $dataDir/MRFcol-Rot-3.out -time -ele 3121 3221 3321 3421
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-1.out -time -ele 4122 4221 force;
#recorder Element -file $dataDir/MRFbeam-Rot-1.out -time -ele 4122 4221
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-2.out -time -ele 4132 4231 force;
#recorder Element -file $dataDir/MRFbeam-Rot-2.out -time -ele 4132 4231
deformation;

DisplayModel2D DeformedShape 1 300 200 1080 720

if {$analysisType == "pushover"} {
        puts "Running Pushover..."
# assign lateral loads and create load pattern:  use ASCE 7-10 distribution
        set lat2 0.0267;        # force on each beam-column joint in Floor 2
        set lat3 0.0556;        # force on each beam-column joint in Floor 3
        set lat4 0.0845;        # force on each beam-column joint in Floor 4
        set lat5 0.1147;        # force on each beam-column joint in Floor 5
        set lat6 0.1446;        # force on each beam-column joint in Floor 6
        set lat7 0.1707;        # force on each beam-column joint in Floor 7
        set lat8 0.1930;        # force on each beam-column joint in Floor 8
        set lat9 0.2101;        # force on each beam-column joint in Floor 9



        pattern Plain 200 Linear {
                                        load 12005 $lat2 0.0 0.0;
                                        #load 2205 $lat2 0.0 0.0;
                                        #load 3205 $lat2 0.0 0.0;
                                        #load 4205 $lat2 0.0 0.0;
                                        load 13005 $lat3 0.0 0.0;
                                        #load 2305 $lat3 0.0 0.0;
                                        #load 3305 $lat3 0.0 0.0;
                                        #load 4305 $lat3 0.0 0.0;
                                        load 14005 $lat4 0.0 0.0;
                                        #load 2405 $lat4 0.0 0.0;
                                        #load 3405 $lat4 0.0 0.0;
                                        #load 4405 $lat4 0.0 0.0;
                                        load 15005 $lat5 0.0 0.0;
                                        #load 2505 $lat5 0.0 0.0;
                                        #load 3505 $lat5 0.0 0.0;
                                        #load 4505 $lat5 0.0 0.0;
                                        load 16005 $lat6 0.0 0.0;
                                        #load 2605 $lat6 0.0 0.0;
                                        #load 3605 $lat6 0.0 0.0;
                                        #load 4605 $lat6 0.0 0.0;
                                        load 17005 $lat7 0.0 0.0;
                                        #load 2705 $lat7 0.0 0.0;
```

```
                                        #load 3705 $lat7 0.0 0.0;
                                        #load 4705 $lat7 0.0 0.0;
                                        load 18005 $lat8 0.0 0.0;
                                        #load 2805 $lat8 0.0 0.0;
                                        #load 3805 $lat8 0.0 0.0;
                                        #load 4805 $lat8 0.0 0.0;
                                        load 19005 $lat9 0.0 0.0;
                                        #load 2905 $lat9 0.0 0.0;
                                        #load 3905 $lat9 0.0 0.0;
                                        #load 4905 $lat9 0.0 0.0;

        }

# display deformed shape:
        #set ViewScale 5;
        #DisplayModel2D DeformedShape $ViewScale ;   # display deformed shape,
the scaling factor needs to be adjusted for each model

# displacement parameters
        set IDctrlNode 19005;                           # node where disp is read
for disp control
        set IDctrlDOF 1;                        # degree of freedom read for disp
control (1 = x displacement)
        set Dmax [expr 0.5*$HBuilding];         # maximum displacement of
pushover: 4% roof drift
        set Dincr [expr 0.05];                  # displacement increment

# analysis commands
        constraints Plain;                      # how it handles boundary
conditions
        numberer Plain;                                 # renumber dof's to
minimize band-width (optimization)
        system UmfPack;                         # how to store and solve the
system of equations in the analysis (large model: try UmfPack)
        set Tol 1.e-4;                          # Convergence Test: tolerance
        set maxNumIter 100;                     # Convergence Test: maximum
number of iterations that will be performed before "failure to converge" is returned
        set printFlag 0;
        set TestType EnergyIncr;

        #test EnergyIncr  1.0e-5 400;           # type of convergence criteria with
tolerance, max iterations
        #test $TestType $Tol $maxNumIter $printFlag;
        test NormUnbalance 0.01 100;

        set algorithmType Newton;
        algorithm $algorithmType;
```

```
	#algorithm Newton;				# use Newton's solution algorithm: updates tangent stiffness at every iteration
	integrator DisplacementControl  $IDctrlNode   $IDctrlDOF $Dincr;	# use displacement-controlled analysis
	analysis Static;				# define type of analysis: static for pushover
	set Nsteps [expr int($Dmax/$Dincr)];# number of pushover analysis steps

	for {set i 1} {$i <= $Nsteps} {incr i 1} {
		set ok [analyze 1];			# this will return zero if no convergence problems were encountered
		puts "PO step $i out of $Nsteps: $ok";
	    if {$ok != 0} {
			source PO_convergence_loop.tcl;
		};
	};



	#set ok [analyze $Nsteps];			# this will return zero if no convergence problems were encountered
	puts "Pushover complete";			# display this message in the command window
}

##########################################################################
#######
#   Time History/Dynamic Analysis                                        #
##########################################################################
#######

if {$analysisType == "dynamic"} {
	puts "Running dynamic analysis..."

	source dynamic.tcl

}

#wipe all;
```

ANEXO L4: CODIGO PARA EJECUTAR LOS ARCHIVOS TCL PRODUCIDOS POR MATLAB EN ESTRUCTURAS DE 12 PISOS

Este Código sirve también para la parte dinámica, para lo cual solo es necesario activar por medio de "#" en la parte que dice "Define análisis Type". Después es necesario desactivar los recorders, de igual manera solo con "#" (al agregarlo al principio de la línea, coloca en comentario la línea)

```
# Element ID conventions:
#       1xy   = frame columns with RBS springs at both ends
#       2xy   = frame beams with RBS springs at both ends
#       6xy   = trusses linking frame and P-delta column
#       7xy   = P-delta columns
#       2xya  = frame beams between panel zone and RBS spring
#       3xya  = frame column rotational springs
#       4xya  = frame beam rotational springs
#       5xya  = P-delta column rotational springs
#       4xy00 = panel zone rotational springs
#       500xya = panel zone elements
#       where:
#               x = Pier or Bay #
#               y = Floor or Story #
#               a = an integer describing the location relative to beam-column joint (see description where elements and nodes are defined)

#########################################################################################
#       Set Up & Source Definition

#########################################################################################
        wipe all;                               # clear memory of past model definitions
        model BasicBuilder -ndm 2 -ndf 3;       # Define the model builder, ndm = #dimension, ndf = #dofs
        source DisplayModel2D.tcl;              # procedure for displaying a 2D perspective of model
        source DisplayPlane.tcl;                # procedure for displaying a plane in a model

#########################################################################################
#       Define Analysis Type
```

```
#########################################################################
##############################

 # Define type of analysis:  "pushover" = pushover;  "dynamic" = dynamic
        #set analysisType "dynamic";
        set analysisType "pushover"



        if {$analysisType == "pushover"} {
                set dataDir Concentrated-PanelZone-Pushover-Output;      # name of
output folder
                file mkdir $dataDir;                                     # create output
folder
        }



#########################################################################
##############################
#       Define Building Geometry, Nodes, Masses, and Constraints

#########################################################################
##############################
        set n 10;

        set NStories 12;                        # number of stories
        set NBays 3;                        # number of frame bays (excludes bay for
P-delta column)
        set WBay      [expr 20.0*12.0];         # bay width in inches
        set HStory1   [expr 15.0*12.0];         # 1st story height in inches
        set HStoryTyp [expr 13.0*12.0];         # story height of other stories in
inches
        set HBuilding [expr $HStory1 + ($NStories-1)*$HStoryTyp];       # height of
building




# assign boundary condidtions
        source Nodes.tcl
        source Constraints.tcl
        # command:  fix nodeID dxFixity dyFixity rzFixity
        # fixity values: 1 = constrained; 0 = unconstrained
        # fix the base of the building; pin P-delta column at base
        fix 11 1 1 1;
        fix 21 1 1 1;
```

```
        fix 31 1 1 1;
        fix 41 1 1 1;
        fix 51 1 1 0;    # P-delta column is pinned


set transfTag 1;
geomTransf PDelta  $transfTag;       # PDelta transformation



        source Material.tcl

        source CreateWSection.tcl
        source Sections.tcl
        source Elements.tcl
        region 4 -ele  111      121     131     141     112     122     132     142
        11391  11392  12391  12392  13391  13392  14391  14392  114     124     134
        144     11591  11592  12591  12592  13591  13592  14591  14592  116     126
        136     146     11791  11792  12791  12792  13791  13792  14791  14792  118
        128     138     148 11391      11992  12991  12992  13991  13992  14991
        14992 1110    1210    1310    1410 111191  111192          121191
        121192          131191          131192          141191          141192 1112   1212
        1312    1412

        source ZeroLengthElem.tcl
        source Masses.tcl




#DisplayModel2D NodeNumbers



##########################################################################
#######
#           Gravity Loads & Gravity Analysis                                    #
##########################################################################
#######

# apply gravity loads
        #command: pattern PatternType $PatternID TimeSeriesType
        pattern Plain 101 Constant {

                # point loads on leaning column nodes
                # command: load node Fx Fy Mz
```

```
set P_PD2 [expr -568.00];     # Floor 2
set P_PD3 [expr -565.20];     # Floor 3
set P_PD13 [expr -511.20];    # Floor 13

load 52 0.0 $P_PD2 0.0;              # Floor 2
load 53 0.0 $P_PD3 0.0;              # Floor 3
load 54 0.0 $P_PD3 0.0;              # Floor 4
load 55 0.0 $P_PD3 0.0;              # Floor 5
load 56 0.0 $P_PD3 0.0;              # Floor 6
load 57 0.0 $P_PD3 0.0;              # Floor 7
load 58 0.0 $P_PD3 0.0;              # Floor 8
load 59 0.0 $P_PD3 0.0;              # Floor 9
load 510 0.0 $P_PD3 0.0;             # Floor 10
load 511 0.0 $P_PD3 0.0;             # Floor 11
load 512 0.0 $P_PD3 0.0;             # Floor 12
load 513 0.0 $P_PD13 0.0;            # Floor 13


# point loads on frame column nodes

set P_F21 [expr -70.21];      # load on each frame node in Floor 2
(exterior)
set P_F22 [expr -46.14];      # load on each frame node in Floor 2
(interior)
set P_F31 [expr -70.21];      # load on each frame node in Floor 3
(exterior)
set P_F32 [expr -46.14];      # load on each frame node in Floor 3
(interior)
set P_F41 [expr -70.21];      # load on each frame node in Floor 4
(exterior)
set P_F42 [expr -46.14];      # load on each frame node in Floor 4
(interior)
set P_F51 [expr -70.21];      # load on each frame node in Floor 5
(exterior)
set P_F52 [expr -46.14];      # load on each frame node in Floor 5
(interior)
set P_F61 [expr -70.21];      # load on each frame node in Floor 6
(exterior)
set P_F62 [expr -46.14];      # load on each frame node in Floor 6
(interior)
set P_F71 [expr -70.21];      # load on each frame node in Floor 7
(exterior)
set P_F72 [expr -46.14];      # load on each frame node in Floor 7
(interior)
set P_F81 [expr -70.21];      # load on each frame node in Floor 8
(exterior)
```

```
set P_F82 [expr -46.14];      # load on each frame node in Floor 8
(interior)
set P_F91 [expr -70.21];      # load on each frame node in Floor 9
(exterior)
set P_F92 [expr -46.14];      # load on each frame node in Floor 9
(interior)
set P_F101 [expr -70.21];     # load on each frame node in Floor 10
(exterior)
set P_F102 [expr -46.14];     # load on each frame node in Floor 10
(interior)
set P_F111 [expr -70.21];     # load on each frame node in Floor 11
(exterior)
set P_F112 [expr -46.14];     # load on each frame node in Floor 11
(interior)
set P_F121 [expr -70.21];     # load on each frame node in Floor 11
(exterior)
set P_F122 [expr -46.14];     # load on each frame node in Floor 11
(interior)
set P_F131 [expr -61.77];     # load on each frame node in Floor 12
(exterior)
set P_F132 [expr -41.18];     # load on each frame node in Floor 12
(interior)


# Floor 2 loads
load 127 0.0 $P_F21 0.0;
load 227 0.0 $P_F22 0.0;
load 327 0.0 $P_F22 0.0;
load 427 0.0 $P_F21 0.0;

# Floor 3 loads
load 137 0.0 $P_F31 0.0;
load 237 0.0 $P_F32 0.0;
load 337 0.0 $P_F32 0.0;
load 437 0.0 $P_F31 0.0;

# Floor 4 loads
load 147 0.0 $P_F41 0.0;
load 247 0.0 $P_F42 0.0;
load 347 0.0 $P_F42 0.0;
load 447 0.0 $P_F41 0.0;

# Floor 5 loads
load 157 0.0 $P_F51 0.0;
load 257 0.0 $P_F52 0.0;
load 357 0.0 $P_F52 0.0;
load 457 0.0 $P_F51 0.0;
```

```
# Floor 6 loads
load 167 0.0 $P_F61 0.0;
load 267 0.0 $P_F62 0.0;
load 367 0.0 $P_F62 0.0;
load 467 0.0 $P_F61 0.0;

# Floor 7 loads
load 177 0.0 $P_F71 0.0;
load 277 0.0 $P_F72 0.0;
load 377 0.0 $P_F72 0.0;
load 477 0.0 $P_F71 0.0;

# Floor 8 loads
load 187 0.0 $P_F81 0.0;
load 287 0.0 $P_F82 0.0;
load 387 0.0 $P_F82 0.0;
load 487 0.0 $P_F81 0.0;

# Floor 9 loads
load 197 0.0 $P_F91 0.0;
load 297 0.0 $P_F92 0.0;
load 397 0.0 $P_F92 0.0;
load 497 0.0 $P_F91 0.0;

# Floor 10 loads
load 1107 0.0 $P_F101 0.0;
load 2107 0.0 $P_F102 0.0;
load 3107 0.0 $P_F102 0.0;
load 4107 0.0 $P_F101 0.0;

# Floor 11 loads
load 1117 0.0 $P_F111 0.0;
load 2117 0.0 $P_F112 0.0;
load 3117 0.0 $P_F112 0.0;
load 4117 0.0 $P_F111 0.0;

# Floor 12 loads
load 1127 0.0 $P_F121 0.0;
load 2127 0.0 $P_F122 0.0;
load 3127 0.0 $P_F122 0.0;
load 4127 0.0 $P_F121 0.0;

# Floor 13 loads
load 1137 0.0 $P_F131 0.0;
load 2137 0.0 $P_F132 0.0;
load 3137 0.0 $P_F132 0.0;
load 4137 0.0 $P_F131 0.0;
```

```
	}

# Gravity-analysis: load-controlled static analysis
	set Tol 1.0e-6;				# convergence tolerance for test
	constraints Plain;			# how it handles boundary conditions
	numberer RCM;					# renumber dof's to minimize
band-width (optimization)
	system BandGeneral;			# how to store and solve the system of
equations in the analysis (large model: try UmfPack)
	test NormDispIncr $Tol 6;		# determine if convergence has been
achieved at the end of an iteration step
	algorithm Newton;			# use Newton's solution algorithm:
updates tangent stiffness at every iteration
	set NstepGravity 1;			# apply gravity in 10 steps
	set DGravity [expr 1.0/$NstepGravity];	# load increment
	integrator LoadControl $DGravity;	# determine the next time step for an
analysis
	analysis Static;			# define type of analysis: static or
transient
	analyze $NstepGravity;			# apply gravity

	# maintain constant gravity loads and reset time to zero
	loadConst -time 0.0
	puts "Model Built"


##########################################################################
#######
#			Eigenvalue Analysis
##########################################################################
#######

	set pi [expr 2.0*asin(1.0)];		# Definition of pi
	set nEigenI 1;					# mode i = 1
	set nEigenJ 2;					# mode j = 2
	set nEigenK 3;					# mode k = 3
	set nEigenL 4;					# mode l = 4
	set nEigenM 5;					# mode m = 5
	set nEigenO 6;					# mode o = 6
	set nEigenP 7;					# mode p = 7
	set nEigenP1 8;					# mode p1 = 8
	set nEigenP2 9;					# mode p2 = 9
	set nEigenP3 10;				# mode p3 = 10
	set nEigenP4 11;				# mode p4 = 11
	set nEigenQ 12;					# mode q = 12
```

```
        set lambdaN [eigen  -genBandArpack  [expr $nEigenQ]];  # eigenvalue
analysis for nEigenQ modes
        set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i = 1
        set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j = 2
        set lambdaK [lindex $lambdaN [expr $nEigenK-1]];        # eigenvalue mode
k = 3
        set lambdaL [lindex $lambdaN [expr $nEigenL-1]];        # eigenvalue mode l
= 4
        set lambdaM [lindex $lambdaN [expr $nEigenM-1]];        # eigenvalue mode
m = 5
        set lambdaO [lindex $lambdaN [expr $nEigenO-1]];        # eigenvalue mode
o = 6
        set lambdaP [lindex $lambdaN [expr $nEigenP-1]];# eigenvalue mode p = 7
        set lambdaP1 [lindex $lambdaN [expr $nEigenP1-1]];      # eigenvalue mode
p1 = 8
        set lambdaP2 [lindex $lambdaN [expr $nEigenP2-1]];      # eigenvalue mode
p2 = 9
        set lambdaP3 [lindex $lambdaN [expr $nEigenP3-1]];      # eigenvalue mode
p3 = 10
        set lambdaP4 [lindex $lambdaN [expr $nEigenP4-1]];      # eigenvalue mode
p4 = 11
        set lambdaQ [lindex $lambdaN [expr $nEigenQ-1]];        # eigenvalue mode
q = 12


        set w1 [expr pow($lambdaI,0.5)];                # w1 (1st mode circular
frequency)
        set w2 [expr pow($lambdaJ,0.5)];                # w2 (2nd mode circular
frequency)
        set w3 [expr pow($lambdaK,0.5)];                # w3 (3st mode circular
frequency)
        set w4 [expr pow($lambdaL,0.5)];                # w4 (4nd mode circular
frequency)
        set w5 [expr pow($lambdaM,0.5)];                # w5 (5st mode circular
frequency)
        set w6 [expr pow($lambdaO,0.5)];                # w6 (6nd mode circular
frequency)
        set w7 [expr pow($lambdaP,0.5)];                # w7 (7st mode circular
frequency)
        set w8 [expr pow($lambdaP1,0.5)];               # w8 (8st mode circular
frequency)
        set w9 [expr pow($lambdaP2,0.5)];               # w9 (9st mode circular
frequency)
        set w10 [expr pow($lambdaP3,0.5)];              # w10 (10st mode circular
frequency)
        set w11 [expr pow($lambdaP4,0.5)];              # w11 (11st mode circular
frequency)
```

```
	set w12 [expr pow($lambdaQ,0.5)];		# w12 (12nd mode circular
frequency)


	set T1 [expr 2.0*$pi/$w1];		# 1st mode period of the
structure
	set T2 [expr 2.0*$pi/$w2];		# 2nd mode period of the
structure
	set T3 [expr 2.0*$pi/$w3];		# 3st mode period of the
structure
	set T4 [expr 2.0*$pi/$w4];		# 4nd mode period of the
structure
	set T5 [expr 2.0*$pi/$w5];		# 5st mode period of the
structure
	set T6 [expr 2.0*$pi/$w6];		# 6nd mode period of the
structure
	set T7 [expr 2.0*$pi/$w7];		# 7st mode period of the
structure
	set T8 [expr 2.0*$pi/$w8];		# 8st mode period of the
structure
	set T9 [expr 2.0*$pi/$w9];		# 9st mode period of the
structure
	set T10 [expr 2.0*$pi/$w10];		# 10st mode period of the
structure
	set T11 [expr 2.0*$pi/$w11];		# 11st mode period of the
structure
	set T12 [expr 2.0*$pi/$w12];		# 12nd mode period of the
structure


	puts "T1 = $T1 s";		# display the first mode
period in the command window
	puts "T2 = $T2 s";		# display the second mode
period in the command window
	puts "T3 = $T3 s";		# display the third mode
period in the command window
	puts "T4 = $T4 s";		# display the fourth mode
period in the command window
	puts "T5 = $T5 s";		# display the fifth mode
period in the command window
	puts "T6 = $T6 s";		# display the sixth mode
period in the command window
	puts "T7 = $T7 s";		# display the seventh mode
period in the command window
	puts "T8 = $T8 s";		# display the eighth mode
period in the command window
	puts "T9 = $T9 s";		# display the ninth mode
period in the command window
```

```
	puts "T10 = $T10 s";				# display the tenth mode
period in the command window
	puts "T11 = $T11 s";				# display the eleventh
mode period in the command window
	puts "T12 = $T12 s";				# display the twelveth
mode period in the command window

##########################################################################
##################
#                                                    #
#                    Analysis Section                                #
#                                                    #
##########################################################################
##################

##########################################################################
#######
#          Pushover Analysis                                  #
##########################################################################
#######

recorder Element -file $dataDir/Fcol_story_forces.out -region 4 force
recorder Node -file $dataDir/Disp_Roof.out -time -node 113005 -dof 1 disp;
#recorder Node -file $dataDir/Rot.out -time -node 11 21 31 41 51 -dof 1 2 3 disp;
#recorder Node -file $dataDir/Vbase.out -time -node 11 21 31 41  -dof 1 2 3 reaction;
#recorder Element -file $dataDir/MRFcol-Mom-1.out -time -ele 3111 3211 3311 3411
force;
#recorder Element -file $dataDir/MRFcol-Rot-1.out -time -ele 3111 3211 3311 3411
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-2.out -time -ele 3112 3212 3312 3412
force;
#recorder Element -file $dataDir/MRFcol-Rot-2.out -time -ele 3112 3212 3312 3412
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-3.out -time -ele 3121 3221 3321 3421
force;
#recorder Element -file $dataDir/MRFcol-Rot-3.out -time -ele 3121 3221 3321 3421
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-1.out -time -ele 4122 4221 force;
#recorder Element -file $dataDir/MRFbeam-Rot-1.out -time -ele 4122 4221
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-2.out -time -ele 4132 4231 force;
#recorder Element -file $dataDir/MRFbeam-Rot-2.out -time -ele 4132 4231
deformation;

DisplayModel2D DeformedShape 1 300 200 1080 720

if {$analysisType == "pushover"} {
	puts "Running Pushover..."
```

```
# assign lateral loads and create load pattern:  use ASCE 7-10 distribution
        set lat2   0.010818357;         # force on each beam-column joint in Floor 2
set lat3   0.02332776; # force on each beam-column joint in Floor 3
set lat4   0.036415441;         # force on each beam-column joint in Floor 4
set lat5   0.050060321;         # force on each beam-column joint in Floor 5
set lat6   0.064337804;         # force on each beam-column joint in Floor 6
set lat7   0.078810471;         # force on each beam-column joint in Floor 7
set lat8   0.092776996;         # force on each beam-column joint in Floor 8
set lat9   0.105902525;         # force on each beam-column joint in Floor 9
set lat10  0.11861572;          # force on each beam-column joint in Floor 10
set lat11  0.130413112;         # force on each beam-column joint in Floor 11
set lat12  0.140506545;         # force on each beam-column joint in Floor 12
set lat13  0.148014947;         # force on each beam-column joint in Floor 13


        pattern Plain 200 Linear {
                                load 12005 $lat2 0.0 0.0;
                                #load 2205 $lat2 0.0 0.0;
                                #load 3205 $lat2 0.0 0.0;
                                #load 4205 $lat2 0.0 0.0;
                                load 13005 $lat3 0.0 0.0;
                                #load 2305 $lat3 0.0 0.0;
                                #load 3305 $lat3 0.0 0.0;
                                #load 4305 $lat3 0.0 0.0;
                                load 14005 $lat4 0.0 0.0;
                                #load 2405 $lat4 0.0 0.0;
                                #load 3405 $lat4 0.0 0.0;
                                #load 4405 $lat4 0.0 0.0;
                                load 15005 $lat5 0.0 0.0;
                                #load 2505 $lat5 0.0 0.0;
                                #load 3505 $lat5 0.0 0.0;
                                #load 4505 $lat5 0.0 0.0;
                                load 16005 $lat6 0.0 0.0;
                                #load 2605 $lat6 0.0 0.0;
                                #load 3605 $lat6 0.0 0.0;
                                #load 4605 $lat6 0.0 0.0;
                                load 17005 $lat7 0.0 0.0;
                                #load 2705 $lat7 0.0 0.0;
                                #load 3705 $lat7 0.0 0.0;
                                #load 4705 $lat7 0.0 0.0;
                                load 18005 $lat8 0.0 0.0;
                                #load 2805 $lat8 0.0 0.0;
                                #load 3805 $lat8 0.0 0.0;
                                #load 4805 $lat8 0.0 0.0;
                                load 19005 $lat9 0.0 0.0;
                                #load 2905 $lat9 0.0 0.0;
                                #load 3905 $lat9 0.0 0.0;
```

```
                                        #load 4905 $lat9 0.0 0.0;
                                        load 110005 $lat10 0.0 0.0;
                                        #load 21005 $lat10 0.0 0.0;
                                        #load 31005 $lat10 0.0 0.0;
                                        #load 41005 $lat10 0.0 0.0;
                                        load 111005 $lat11 0.0 0.0;
                                        #load 21105 $lat11 0.0 0.0;
                                        #load 31105 $lat11 0.0 0.0;
                                        #load 41105 $lat11 0.0 0.0;
                                        load 112005 $lat12 0.0 0.0;
                                        #load 21205 $lat12 0.0 0.0;
                                        #load 31205 $lat12 0.0 0.0;
                                        #load 41205 $lat12 0.0 0.0;
                                        load 113005 $lat13 0.0 0.0;
                                        #load 21305 $lat13 0.0 0.0;
                                        #load 31305 $lat13 0.0 0.0;
                                        #load 41305 $lat13 0.0 0.0;

        }

# display deformed shape:
        set ViewScale 5;
        #DisplayModel2D DeformedShape $ViewScale ;   # display deformed shape,
the scaling factor needs to be adjusted for each model

# displacement parameters
        set IDctrlNode 113005;                          # node where disp is read
for disp control
        set IDctrlDOF 1;                        # degree of freedom read for disp
control (1 = x displacement)
        set Dmax [expr 0.4*$HBuilding];         # maximum displacement of
pushover: 4% roof drift
        set Dincr [expr 0.1];                   # displacement increment

# analysis commands
        constraints Plain;                      # how it handles boundary
conditions
        numberer Plain;                                 # renumber dof's to
minimize band-width (optimization)
        system UmfPack;                         # how to store and solve the
system of equations in the analysis (large model: try UmfPack)
        set Tol 1.e-4;                          # Convergence Test: tolerance
        set maxNumIter 100;                     # Convergence Test: maximum
number of iterations that will be performed before "failure to converge" is returned
        set printFlag 0;
        set TestType EnergyIncr;
```

```
	#test EnergyIncr  1.0e-5 400;			# type of convergence criteria with
tolerance, max iterations
	#test $TestType $Tol $maxNumIter $printFlag;
	test NormUnbalance 0.01 100;

	set algorithmType Newton;
	algorithm $algorithmType;


	#algorithm Newton;				# use Newton's solution algorithm:
updates tangent stiffness at every iteration
	integrator DisplacementControl  $IDctrlNode   $IDctrlDOF $Dincr;	# use
displacement-controlled analysis
	analysis Static;				# define type of analysis: static for
pushover
	set Nsteps [expr int($Dmax/$Dincr)];# number of pushover analysis steps

	for {set i 1} {$i <= $Nsteps} {incr i 1} {
		set ok [analyze 1];			# this will return zero if no
convergence problems were encountered
		puts "PO step $i out of $Nsteps: $ok";
	    if {$ok != 0} {
			source PO_convergence_loop.tcl;
		};
	};




	#set ok [analyze $Nsteps];			# this will return zero if no
convergence problems were encountered
	puts "Pushover complete";			# display this message in the
command window
}

##########################################################################
#######
#   Time History/Dynamic Analysis                                        #
##########################################################################
#######

if {$analysisType == "dynamic"} {
	puts "Running dynamic analysis..."
	source dynamic.tcl

}

#wipe all;
```

ANEXO L5: CODIGO PARA EJECUTAR LOS ARCHIVOS TCL PRODUCIDOS POR MATLAB EN ESTRUCTURAS DE 20 PISOS

Este Código sirve también para la parte dinámica, para lo cual solo es necesario activar por medio de "#" en la parte que dice "Define análisis Type". Después es necesario desactivar los recorders, de igual manera solo con "#" (al agregarlo al principio de la línea, coloca en comentario la línea)

```
# Element ID conventions:
#       1xy    = frame columns with RBS springs at both ends
#       2xy    = frame beams with RBS springs at both ends
#       6xy    = trusses linking frame and P-delta column
#       7xy    = P-delta columns
#       2xya   = frame beams between panel zone and RBS spring
#       3xya   = frame column rotational springs
#       4xya   = frame beam rotational springs
#       5xya   = P-delta column rotational springs
#       4xy00  = panel zone rotational springs
#       500xya = panel zone elements
#       where:
#               x = Pier or Bay #
#               y = Floor or Story #
#               a = an integer describing the location relative to beam-column joint (see description where elements and nodes are defined)

#############################################################################################
#       Set Up & Source Definition

#############################################################################################
        wipe all;                               # clear memory of past model definitions
        model BasicBuilder -ndm 2 -ndf 3;       # Define the model builder, ndm = #dimension, ndf = #dofs
        source DisplayModel2D.tcl;              # procedure for displaying a 2D perspective of model
        source DisplayPlane.tcl;                # procedure for displaying a plane in a model

#############################################################################################
#       Define Analysis Type

#############################################################################################
```

```
 # Define type of analysis:  "pushover" = pushover;  "dynamic" = dynamic
        #set analysisType "dynamic";
        set analysisType "pushover"



        if {$analysisType == "pushover"} {
                set dataDir Concentrated-PanelZone-Pushover-Output;      # name of output
folder
                file mkdir $dataDir;                                     # create output folder
        }



##########################################################################
########################
#       Define Building Geometry, Nodes, Masses, and Constraints

##########################################################################
########################
        set n 10;

        set NStories 20;                        # number of stories
        set NBays 3;                    # number of frame bays (excludes bay for P-delta
column)
        set WBay      [expr 20.0*12.0];         # bay width in inches
        set HStory1   [expr 15.0*12.0];         # 1st story height in inches
        set HStoryTyp [expr 13.0*12.0];         # story height of other stories in inches
        set HBuilding [expr $HStory1 + ($NStories-1)*$HStoryTyp];      # height of building




# assign boundary condidtions
        source Nodes.tcl
        source Constraints.tcl
        # command:  fix nodeID dxFixity dyFixity rzFixity
        # fixity values: 1 = constrained; 0 = unconstrained
        # fix the base of the building; pin P-delta column at base
        fix 11 1 1 1;
        fix 21 1 1 1;
        fix 31 1 1 1;
        fix 41 1 1 1;
        fix 51 1 1 0;   # P-delta column is pinned
```

```
set transfTag 1;
geomTransf PDelta  $transfTag;       # PDelta transformation



        source Material.tcl

        source CreateWSection.tcl
        source Sections.tcl
        source Elements.tcl
        region 4 -ele  111      121     131     141     112     122     132     142     11391
        11392  12391  12392  13391  13392  14391  14392  114     124     134     144
        11591  11592  12591  12592  13591  13592  14591  14592  116     126     136     146
        11791  11792  12791  12792  13791  13792  14791  14792  118     128     138     148
11391  11992  12991  12992  13991  13992  14991  14992 1110    1210    1310    1410 111191
        111192          121191          121192          131191          131192          141191
        141192 1112   1212    1312    1412 111391   111392          121391          121392
        131391          131392          141391          141392 1114   1214    1314    1414 111591
        111592          121591          121592          131591          131592          141591
        141592 1116   1216    1316    1416 111791   111792          121791          121792
        131791          131792          141791          141792 1118   1218    1318    1418 111991
        111992          121991          121992          131991          131992          141991
        141992 1120   1220    1320    1420

        source ZeroLengthElem.tcl
        source Masses.tcl




#DisplayModel2D NodeNumbers



##############################################################################
#
#           Gravity Loads & Gravity Analysis                                 #
##############################################################################
#

# apply gravity loads
        #command: pattern PatternType $PatternID TimeSeriesType
        pattern Plain 101 Constant {

                # point loads on leaning column nodes
                # command: load node Fx Fy Mz
```

```
set P_PD2 [expr -568.00];     # Floor 2
set P_PD3 [expr -565.20];     # Floor 3
set P_PD21 [expr -511.20];    # Floor 21

load 52 0.0 $P_PD2 0.0;               # Floor 2
load 53 0.0 $P_PD3 0.0;               # Floor 3
load 54 0.0 $P_PD3 0.0;               # Floor 4
load 55 0.0 $P_PD3 0.0;               # Floor 5
load 56 0.0 $P_PD3 0.0;               # Floor 6
load 57 0.0 $P_PD3 0.0;               # Floor 7
load 58 0.0 $P_PD3 0.0;               # Floor 8
load 59 0.0 $P_PD3 0.0;               # Floor 9
load 510 0.0 $P_PD3 0.0;              # Floor 10
load 511 0.0 $P_PD3 0.0;              # Floor 11
load 512 0.0 $P_PD3 0.0;              # Floor 12
load 513 0.0 $P_PD3 0.0;              # Floor 13
load 514 0.0 $P_PD3 0.0;              # Floor 14
load 515 0.0 $P_PD3 0.0;              # Floor 15
load 516 0.0 $P_PD3 0.0;              # Floor 16
load 517 0.0 $P_PD3 0.0;              # Floor 17
load 518 0.0 $P_PD3 0.0;              # Floor 18
load 519 0.0 $P_PD3 0.0;              # Floor 19
load 520 0.0 $P_PD3 0.0;              # Floor 20
load 521 0.0 $P_PD21 0.0;     # Floor 21



# point loads on frame column nodes


set P_F21 [expr -70.21];      # load on each frame node in Floor 2 (exterior)
set P_F22 [expr -46.14];      # load on each frame node in Floor 2 (interior)
set P_F31 [expr -70.21];      # load on each frame node in Floor 3 (exterior)
set P_F32 [expr -46.14];      # load on each frame node in Floor 3 (interior)
set P_F41 [expr -70.21];      # load on each frame node in Floor 4 (exterior)
set P_F42 [expr -46.14];      # load on each frame node in Floor 4 (interior)
set P_F51 [expr -70.21];      # load on each frame node in Floor 5 (exterior)
set P_F52 [expr -46.14];      # load on each frame node in Floor 5 (interior)
set P_F61 [expr -70.21];      # load on each frame node in Floor 6 (exterior)
set P_F62 [expr -46.14];      # load on each frame node in Floor 6 (interior)
set P_F71 [expr -70.21];      # load on each frame node in Floor 7 (exterior)
set P_F72 [expr -46.14];      # load on each frame node in Floor 7 (interior)
set P_F81 [expr -70.21];      # load on each frame node in Floor 8 (exterior)
set P_F82 [expr -46.14];      # load on each frame node in Floor 8 (interior)
set P_F91 [expr -70.21];      # load on each frame node in Floor 9 (exterior)
set P_F92 [expr -46.14];      # load on each frame node in Floor 9 (interior)
set P_F101 [expr -70.21];     # load on each frame node in Floor 10 (exterior)
set P_F102 [expr -46.14];     # load on each frame node in Floor 10 (interior)
```

```
set P_F111 [expr -70.21];     # load on each frame node in Floor 11 (exterior)
set P_F112 [expr -46.14];     # load on each frame node in Floor 11 (interior)
set P_F121 [expr -70.21];     # load on each frame node in Floor 12 (exterior)
set P_F122 [expr -46.14];     # load on each frame node in Floor 12 (interior)
set P_F131 [expr -70.21];     # load on each frame node in Floor 13 (exterior)
set P_F132 [expr -46.14];     # load on each frame node in Floor 13 (interior)
set P_F141 [expr -70.21];     # load on each frame node in Floor 14 (exterior)
set P_F142 [expr -46.14];     # load on each frame node in Floor 14 (interior)
set P_F151 [expr -70.21];     # load on each frame node in Floor 15 (exterior)
set P_F152 [expr -46.14];     # load on each frame node in Floor 15 (interior)
set P_F161 [expr -70.21];     # load on each frame node in Floor 16 (exterior)
set P_F162 [expr -46.14];     # load on each frame node in Floor 16 (interior)
set P_F171 [expr -70.21];     # load on each frame node in Floor 17 (exterior)
set P_F172 [expr -46.14];     # load on each frame node in Floor 17 (interior)
set P_F181 [expr -70.21];     # load on each frame node in Floor 18 (exterior)
set P_F182 [expr -46.14];     # load on each frame node in Floor 18 (interior)
set P_F191 [expr -70.21];     # load on each frame node in Floor 19 (exterior)
set P_F192 [expr -46.14];     # load on each frame node in Floor 19 (interior)
set P_F201 [expr -70.21];     # load on each frame node in Floor 20 (exterior)
set P_F202 [expr -46.14];     # load on each frame node in Floor 20 (interior)
set P_F211 [expr -61.77];     # load on each frame node in Floor 21 (exterior)
set P_F212 [expr -41.18];     # load on each frame node in Floor 21 (interior)


# Floor 2 loads
load 127 0.0 $P_F21 0.0;
load 227 0.0 $P_F22 0.0;
load 327 0.0 $P_F22 0.0;
load 427 0.0 $P_F21 0.0;

# Floor 3 loads
load 137 0.0 $P_F31 0.0;
load 237 0.0 $P_F32 0.0;
load 337 0.0 $P_F32 0.0;
load 437 0.0 $P_F31 0.0;

# Floor 4 loads
load 147 0.0 $P_F41 0.0;
load 247 0.0 $P_F42 0.0;
load 347 0.0 $P_F42 0.0;
load 447 0.0 $P_F41 0.0;

# Floor 5 loads
load 157 0.0 $P_F51 0.0;
load 257 0.0 $P_F52 0.0;
load 357 0.0 $P_F52 0.0;
load 457 0.0 $P_F51 0.0;
```

```
# Floor 6 loads
load 167 0.0 $P_F61 0.0;
load 267 0.0 $P_F62 0.0;
load 367 0.0 $P_F62 0.0;
load 467 0.0 $P_F61 0.0;

# Floor 7 loads
load 177 0.0 $P_F71 0.0;
load 277 0.0 $P_F72 0.0;
load 377 0.0 $P_F72 0.0;
load 477 0.0 $P_F71 0.0;

# Floor 8 loads
load 187 0.0 $P_F81 0.0;
load 287 0.0 $P_F82 0.0;
load 387 0.0 $P_F82 0.0;
load 487 0.0 $P_F81 0.0;

# Floor 9 loads
load 197 0.0 $P_F91 0.0;
load 297 0.0 $P_F92 0.0;
load 397 0.0 $P_F92 0.0;
load 497 0.0 $P_F91 0.0;

# Floor 10 loads
load 1107 0.0 $P_F101 0.0;
load 2107 0.0 $P_F102 0.0;
load 3107 0.0 $P_F102 0.0;
load 4107 0.0 $P_F101 0.0;

# Floor 11 loads
load 1117 0.0 $P_F111 0.0;
load 2117 0.0 $P_F112 0.0;
load 3117 0.0 $P_F112 0.0;
load 4117 0.0 $P_F111 0.0;

# Floor 12 loads
load 1127 0.0 $P_F121 0.0;
load 2127 0.0 $P_F122 0.0;
load 3127 0.0 $P_F122 0.0;
load 4127 0.0 $P_F121 0.0;

# Floor 13 loads
load 1137 0.0 $P_F131 0.0;
load 2137 0.0 $P_F132 0.0;
load 3137 0.0 $P_F132 0.0;
load 4137 0.0 $P_F131 0.0;
```

```
# Floor 14 loads
load 1147 0.0 $P_F141 0.0;
load 2147 0.0 $P_F142 0.0;
load 3147 0.0 $P_F142 0.0;
load 4147 0.0 $P_F141 0.0;

# Floor 15 loads
load 1157 0.0 $P_F151 0.0;
load 2157 0.0 $P_F152 0.0;
load 3157 0.0 $P_F152 0.0;
load 4157 0.0 $P_F151 0.0;

# Floor 16 loads
load 1167 0.0 $P_F161 0.0;
load 2167 0.0 $P_F162 0.0;
load 3167 0.0 $P_F162 0.0;
load 4167 0.0 $P_F161 0.0;

# Floor 17 loads
load 1177 0.0 $P_F171 0.0;
load 2177 0.0 $P_F172 0.0;
load 3177 0.0 $P_F172 0.0;
load 4177 0.0 $P_F171 0.0;

# Floor 18 loads
load 1187 0.0 $P_F181 0.0;
load 2187 0.0 $P_F182 0.0;
load 3187 0.0 $P_F182 0.0;
load 4187 0.0 $P_F181 0.0;

# Floor 19 loads
load 1197 0.0 $P_F191 0.0;
load 2197 0.0 $P_F192 0.0;
load 3197 0.0 $P_F192 0.0;
load 4197 0.0 $P_F191 0.0;

# Floor 20 loads
load 1207 0.0 $P_F201 0.0;
load 2207 0.0 $P_F202 0.0;
load 3207 0.0 $P_F202 0.0;
load 4207 0.0 $P_F201 0.0;

# Floor 21 loads
load 1217 0.0 $P_F211 0.0;
load 2217 0.0 $P_F212 0.0;
load 3217 0.0 $P_F212 0.0;
load 4217 0.0 $P_F211 0.0;
```

```
	}

# Gravity-analysis: load-controlled static analysis
	set Tol 1.0e-6;				# convergence tolerance for test
	constraints Plain;			# how it handles boundary conditions
	numberer RCM;					# renumber dof's to minimize band-width
(optimization)
	system BandGeneral;			# how to store and solve the system of equations
in the analysis (large model: try UmfPack)
	test NormDispIncr $Tol 6;		# determine if convergence has been achieved at
the end of an iteration step
	algorithm Newton;			# use Newton's solution algorithm: updates
tangent stiffness at every iteration
	set NstepGravity 1;			# apply gravity in 10 steps
	set DGravity [expr 1.0/$NstepGravity];	# load increment
	integrator LoadControl $DGravity;	# determine the next time step for an analysis
	analysis Static;			# define type of analysis: static or transient
	analyze $NstepGravity;			# apply gravity

	# maintain constant gravity loads and reset time to zero
	loadConst -time 0.0
	puts "Model Built"


##########################################################################################
#
#			Eigenvalue Analysis
##########################################################################################
#

	set pi [expr 2.0*asin(1.0)];		# Definition of pi
	set nEigenI 1;					# mode i = 1
	set nEigenJ 2;					# mode j = 2
	set nEigenK 3;						# mode k = 3
	set nEigenL 4;					# mode l = 4
	set nEigenM 5;						# mode m = 5
	set nEigenO 6;						# mode o = 6
	set nEigenP 7;					# mode p = 7
	set nEigenP1 8;						# mode p1 = 8
	set nEigenP2 9;						# mode p2 = 9
	set nEigenP3 10;				# mode p3 = 10
	set nEigenP4 11;				# mode p4 = 11
	set nEigenP5 12;				# mode p5 = 12
	set nEigenP6 13;				# mode p6 = 13
	set nEigenP7 14;				# mode p7 = 14
```

```
	set nEigenP8 15;				# mode p8 = 15
	set nEigenP9 16;				# mode p9 = 16
	set nEigenP10 17;				# mode p10 = 17
	set nEigenP11 18;				# mode p11 = 18
	set nEigenP12 19;				# mode p12 = 19
	set nEigenQ 20;					# mode q = 20

	set lambdaN [eigen  -genBandArpack  [expr $nEigenQ]];  # eigenvalue analysis for
nEigenQ modes
	set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i = 1
	set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j = 2
	set lambdaK [lindex $lambdaN [expr $nEigenK-1]];	# eigenvalue mode k = 3
	set lambdaL [lindex $lambdaN [expr $nEigenL-1]];	# eigenvalue mode l = 4
	set lambdaM [lindex $lambdaN [expr $nEigenM-1]];	# eigenvalue mode m = 5
	set lambdaO [lindex $lambdaN [expr $nEigenO-1]];	# eigenvalue mode o = 6
	set lambdaP [lindex $lambdaN [expr $nEigenP-1]];# eigenvalue mode p = 7
	set lambdaP1 [lindex $lambdaN [expr $nEigenP1-1]];	# eigenvalue mode p1 = 8
	set lambdaP2 [lindex $lambdaN [expr $nEigenP2-1]];	# eigenvalue mode p2 = 9
	set lambdaP3 [lindex $lambdaN [expr $nEigenP3-1]];	# eigenvalue mode p3 = 10
	set lambdaP4 [lindex $lambdaN [expr $nEigenP4-1]];	# eigenvalue mode p4 = 11
	set lambdaP5 [lindex $lambdaN [expr $nEigenP5-1]];	# eigenvalue mode p5 = 12
	set lambdaP6 [lindex $lambdaN [expr $nEigenP6-1]];	# eigenvalue mode p6 = 13
	set lambdaP7 [lindex $lambdaN [expr $nEigenP7-1]];	# eigenvalue mode p7 = 14
	set lambdaP8 [lindex $lambdaN [expr $nEigenP8-1]];	# eigenvalue mode p8 = 15
	set lambdaP9 [lindex $lambdaN [expr $nEigenP9-1]];	# eigenvalue mode p9 = 16
	set lambdaP10 [lindex $lambdaN [expr $nEigenP10-1]];	# eigenvalue mode p10 =
17
	set lambdaP11 [lindex $lambdaN [expr $nEigenP11-1]];	# eigenvalue mode p11 =
18
	set lambdaP12 [lindex $lambdaN [expr $nEigenP12-1]];	# eigenvalue mode p12 =
19
	set lambdaQ [lindex $lambdaN [expr $nEigenQ-1]];	# eigenvalue mode q = 20


	set w1 [expr pow($lambdaI,0.5)];			# w1 (1st mode circular frequency)
	set w2 [expr pow($lambdaJ,0.5)];			# w2 (2nd mode circular
frequency)
	set w3 [expr pow($lambdaK,0.5)];			# w3 (3rd mode circular
frequency)
	set w4 [expr pow($lambdaL,0.5)];			# w4 (4th mode circular
frequency)
	set w5 [expr pow($lambdaM,0.5)];			# w5 (5th mode circular
frequency)
	set w6 [expr pow($lambdaO,0.5)];			# w6 (6th mode circular
frequency)
	set w7 [expr pow($lambdaP,0.5)];			# w7 (7th mode circular
frequency)
```

```
	set w8 [expr pow($lambdaP1,0.5)];		# w8 (8th mode circular
frequency)
	set w9 [expr pow($lambdaP2,0.5)];		# w9 (9th mode circular
frequency)
	set w10 [expr pow($lambdaP3,0.5)];		# w10 (10th mode circular
frequency)
	set w11 [expr pow($lambdaP4,0.5)];		# w11 (11th mode circular
frequency)
	set w12 [expr pow($lambdaP5,0.5)];		# w12 (12th mode circular
frequency)
	set w13 [expr pow($lambdaP6,0.5)];		# w13 (13th mode circular
frequency)
	set w14 [expr pow($lambdaP7,0.5)];		# w14 (14th mode circular
frequency)
	set w15 [expr pow($lambdaP8,0.5)];		# w15 (15th mode circular
frequency)
	set w16 [expr pow($lambdaP9,0.5)];		# w16 (16th mode circular
frequency)
	set w17 [expr pow($lambdaP10,0.5)];		# w17 (17th mode circular
frequency)
	set w18 [expr pow($lambdaP11,0.5)];		# w18 (18th mode circular
frequency)
	set w19 [expr pow($lambdaP12,0.5)];		# w19 (19th mode circular
frequency)
	set w20 [expr pow($lambdaQ,0.5)];		# w20 (20th mode circular
frequency)


	set T1 [expr 2.0*$pi/$w1];		# 1st mode period of the structure
	set T2 [expr 2.0*$pi/$w2];		# 2nd mode period of the structure
	set T3 [expr 2.0*$pi/$w3];		# 3rd mode period of the structure
	set T4 [expr 2.0*$pi/$w4];		# 4th mode period of the structure
	set T5 [expr 2.0*$pi/$w5];		# 5th mode period of the structure
	set T6 [expr 2.0*$pi/$w6];		# 6th mode period of the structure
	set T7 [expr 2.0*$pi/$w7];		# 7th mode period of the structure
	set T8 [expr 2.0*$pi/$w8];		# 8th mode period of the structure
	set T9 [expr 2.0*$pi/$w9];		# 9th mode period of the structure
	set T10 [expr 2.0*$pi/$w10];		# 10th mode period of the structure
	set T11 [expr 2.0*$pi/$w11];		# 11th mode period of the structure
	set T12 [expr 2.0*$pi/$w12];		# 12th mode period of the structure
	set T13 [expr 2.0*$pi/$w13];		# 13th mode period of the structure
	set T14 [expr 2.0*$pi/$w14];		# 14th mode period of the structure
	set T15 [expr 2.0*$pi/$w15];		# 15th mode period of the structure
	set T16 [expr 2.0*$pi/$w16];		# 16th mode period of the structure
	set T17 [expr 2.0*$pi/$w17];		# 17th mode period of the structure
	set T18 [expr 2.0*$pi/$w18];		# 18th mode period of the structure
	set T19 [expr 2.0*$pi/$w19];		# 19th mode period of the structure
	set T20 [expr 2.0*$pi/$w20];		# 20th mode period of the structure
```

```
	puts "T1 = $T1 s";				# display the first mode period in
the command window
	puts "T2 = $T2 s";				# display the second mode period
in the command window
	puts "T3 = $T3 s";				# display the third mode period in
the command window
	puts "T4 = $T4 s";				# display the fourth mode period in
the command window
	puts "T5 = $T5 s";				# display the fifth mode period in
the command window
	puts "T6 = $T6 s";				# display the sixth mode period in
the command window
	puts "T7 = $T7 s";				# display the seventh mode period
in the command window
	puts "T8 = $T8 s";				# display the eighth mode period
in the command window
	puts "T9 = $T9 s";				# display the ninth mode period in
the command window
	puts "T10 = $T10 s";				# display the tenth mode period in
the command window
	puts "T11 = $T11 s";				# display the eleventh mode period
in the command window
	puts "T12 = $T12 s";				# display the twelveth mode period
in the command window
	puts "T13 = $T13 s";				# display the thirteenth mode
period in the command window
	puts "T14 = $T14 s";				# display the fourteenth mode
period in the command window
	puts "T15 = $T15 s";				# display the fifteenth mode period
in the command window
	puts "T16 = $T16 s";				# display the sixteenth mode
period in the command window
	puts "T17 = $T17 s";				# display the seventeenth mode
period in the command window
	puts "T18 = $T18 s";				# display the eighteenth mode
period in the command window
	puts "T19 = $T19 s";				# display the nineteenth mode
period in the command window
	puts "T20 = $T20 s";				# display the twentieth mode
period in the command window

################################################################################
############
#                                                #
#                Analysis Section                        #
#                                                #
```

```
##########################################################################################
############

##########################################################################################
#
#            Pushover Analysis                                        #
##########################################################################################
#

recorder Element -file $dataDir/Fcol_story_forces.out -region 4 force
recorder Node -file $dataDir/Disp_Roof.out -time -node 121005 -dof 1 disp;
#recorder Node -file $dataDir/Rot.out -time -node 11 21 31 41 51 -dof 1 2 3 disp;
#recorder Node -file $dataDir/Vbase.out -time -node 11 21 31 41  -dof 1 2 3 reaction;
#recorder Element -file $dataDir/MRFcol-Mom-1.out -time -ele 3111 3211 3311 3411 force;
#recorder Element -file $dataDir/MRFcol-Rot-1.out -time -ele 3111 3211 3311 3411
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-2.out -time -ele 3112 3212 3312 3412 force;
#recorder Element -file $dataDir/MRFcol-Rot-2.out -time -ele 3112 3212 3312 3412
deformation;
#recorder Element -file $dataDir/MRFcol-Mom-3.out -time -ele 3121 3221 3321 3421 force;
#recorder Element -file $dataDir/MRFcol-Rot-3.out -time -ele 3121 3221 3321 3421
deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-1.out -time -ele 4122 4221 force;
#recorder Element -file $dataDir/MRFbeam-Rot-1.out -time -ele 4122 4221 deformation;
#recorder Element -file $dataDir/MRFbeam-Mom-2.out -time -ele 4132 4231 force;
#recorder Element -file $dataDir/MRFbeam-Rot-2.out -time -ele 4132 4231 deformation;

DisplayModel2D DeformedShape 1 300 200 1080 720

if {$analysisType == "pushover"} {
        puts "Running Pushover..."
# assign lateral loads and create load pattern:  use ASCE 7-10 distribution
        set lat2 0.0038;        # force on each beam-column joint in Floor 2
set lat3 0.0080;        # force on each beam-column joint in Floor 3
set lat4 0.0124;        # force on each beam-column joint in Floor 4
set lat5 0.0170;        # force on each beam-column joint in Floor 5
set lat6 0.0217;        # force on each beam-column joint in Floor 6
set lat7 0.0264;        # force on each beam-column joint in Floor 7
set lat8 0.0312;        # force on each beam-column joint in Floor 8
set lat9 0.0361;        # force on each beam-column joint in Floor 9
set lat10 0.0413;       # force on each beam-column joint in Floor 10
set lat11 0.0468;       # force on each beam-column joint in Floor 11
set lat12 0.0522;       # force on each beam-column joint in Floor 12
set lat13 0.0575;       # force on each beam-column joint in Floor 13
set lat14 0.0626;       # force on each beam-column joint in Floor 14
set lat15 0.0676;       # force on each beam-column joint in Floor 15
set lat16 0.0732;       # force on each beam-column joint in Floor 16
set lat17 0.0789;       # force on each beam-column joint in Floor 17
```

```
set lat18 0.0841;       # force on each beam-column joint in Floor 18
set lat19 0.0886;       # force on each beam-column joint in Floor 19
set lat20 0.0932;       # force on each beam-column joint in Floor 20
set lat21 0.0973;       # force on each beam-column joint in Floor 21


        pattern Plain 200 Linear {
                                        load 12005 $lat2 0.0 0.0;
                                        #load 2205 $lat2 0.0 0.0;
                                        #load 3205 $lat2 0.0 0.0;
                                        #load 4205 $lat2 0.0 0.0;
                                        load 13005 $lat3 0.0 0.0;
                                        #load 2305 $lat3 0.0 0.0;
                                        #load 3305 $lat3 0.0 0.0;
                                        #load 4305 $lat3 0.0 0.0;
                                        load 14005 $lat4 0.0 0.0;
                                        #load 2405 $lat4 0.0 0.0;
                                        #load 3405 $lat4 0.0 0.0;
                                        #load 4405 $lat4 0.0 0.0;
                                        load 15005 $lat5 0.0 0.0;
                                        #load 2505 $lat5 0.0 0.0;
                                        #load 3505 $lat5 0.0 0.0;
                                        #load 4505 $lat5 0.0 0.0;
                                        load 16005 $lat6 0.0 0.0;
                                        #load 2605 $lat6 0.0 0.0;
                                        #load 3605 $lat6 0.0 0.0;
                                        #load 4605 $lat6 0.0 0.0;
                                        load 17005 $lat7 0.0 0.0;
                                        #load 2705 $lat7 0.0 0.0;
                                        #load 3705 $lat7 0.0 0.0;
                                        #load 4705 $lat7 0.0 0.0;
                                        load 18005 $lat8 0.0 0.0;
                                        #load 2805 $lat8 0.0 0.0;
                                        #load 3805 $lat8 0.0 0.0;
                                        #load 4805 $lat8 0.0 0.0;
                                        load 19005 $lat9 0.0 0.0;
                                        #load 2905 $lat9 0.0 0.0;
                                        #load 3905 $lat9 0.0 0.0;
                                        #load 4905 $lat9 0.0 0.0;
                                        load 110005 $lat10 0.0 0.0;
                                        #load 21005 $lat10 0.0 0.0;
                                        #load 31005 $lat10 0.0 0.0;
                                        #load 41005 $lat10 0.0 0.0;
                                        load 111005 $lat11 0.0 0.0;
                                        #load 21105 $lat11 0.0 0.0;
                                        #load 31105 $lat11 0.0 0.0;
                                        #load 41105 $lat11 0.0 0.0;
                                        load 112005 $lat12 0.0 0.0;
```

```
#load 21205 $lat12 0.0 0.0;
#load 31205 $lat12 0.0 0.0;
#load 41205 $lat12 0.0 0.0;
load 113005 $lat13 0.0 0.0;
#load 21305 $lat13 0.0 0.0;
#load 31305 $lat13 0.0 0.0;
#load 41305 $lat13 0.0 0.0;
load 114005 $lat14 0.0 0.0;
#load 21405 $lat14 0.0 0.0;
#load 31405 $lat14 0.0 0.0;
#load 41405 $lat14 0.0 0.0;
load 115005 $lat15 0.0 0.0;
#load 21505 $lat15 0.0 0.0;
#load 31505 $lat15 0.0 0.0;
#load 41505 $lat15 0.0 0.0;
load 116005 $lat16 0.0 0.0;
#load 21605 $lat16 0.0 0.0;
#load 31605 $lat16 0.0 0.0;
#load 41605 $lat16 0.0 0.0;
load 117005 $lat17 0.0 0.0;
#load 21705 $lat17 0.0 0.0;
#load 31705 $lat17 0.0 0.0;
#load 41705 $lat17 0.0 0.0;
load 118005 $lat18 0.0 0.0;
#load 21805 $lat18 0.0 0.0;
#load 31805 $lat18 0.0 0.0;
#load 41805 $lat18 0.0 0.0;
load 119005 $lat19 0.0 0.0;
#load 21905 $lat19 0.0 0.0;
#load 31905 $lat19 0.0 0.0;
#load 41905 $lat19 0.0 0.0;
load 120005 $lat20 0.0 0.0;
#load 22005 $lat20 0.0 0.0;
#load 32005 $lat20 0.0 0.0;
#load 42005 $lat20 0.0 0.0;
load 121005 $lat21 0.0 0.0;
#load 22105 $lat21 0.0 0.0;
#load 32105 $lat21 0.0 0.0;
#load 42105 $lat21 0.0 0.0;

        }

# display deformed shape:
        set ViewScale 5;
        #DisplayModel2D DeformedShape $ViewScale ;    # display deformed shape, the
scaling factor needs to be adjusted for each model

# displacement parameters
```

```
	set IDctrlNode 121005;				# node where disp is read for disp
control
	set IDctrlDOF 1;				# degree of freedom read for disp control
(1 = x displacement)
	set Dmax [expr 0.5*$HBuilding];		# maximum displacement of pushover:
4% roof drift
	set Dincr [expr 0.05];				# displacement increment

# analysis commands
	constraints Plain;				# how it handles boundary conditions
	numberer Plain;					# renumber dof's to minimize
band-width (optimization)
	system UmfPack;				# how to store and solve the system of
equations in the analysis (large model: try UmfPack)
	set Tol 1.e-4;				# Convergence Test: tolerance
	set maxNumIter 100;				# Convergence Test: maximum number of
iterations that will be performed before "failure to converge" is returned
	set printFlag 0;
	set TestType EnergyIncr;

	#test EnergyIncr  1.0e-5 400;		# type of convergence criteria with
tolerance, max iterations
	#test $TestType $Tol $maxNumIter $printFlag;
	test NormUnbalance 0.01 100;

	set algorithmType Newton;
	algorithm $algorithmType;


	#algorithm Newton;				# use Newton's solution algorithm:
updates tangent stiffness at every iteration
	integrator DisplacementControl  $IDctrlNode   $IDctrlDOF $Dincr;	# use
displacement-controlled analysis
	analysis Static;				# define type of analysis: static for
pushover
	set Nsteps [expr int($Dmax/$Dincr)];# number of pushover analysis steps

	for {set i 1} {$i <= $Nsteps} {incr i 1} {
		set ok [analyze 1];			# this will return zero if no convergence
problems were encountered
		puts "PO step $i out of $Nsteps: $ok";
	    if {$ok != 0} {
			source PO_convergence_loop.tcl;
		};
	};
```

```
	#set ok [analyze $Nsteps];			# this will return zero if no convergence
problems were encountered
	puts "Pushover complete";			# display this message in the command
window
}

##########################################################################
#
#   Time History/Dynamic Analysis                                        #
##########################################################################
#

if {$analysisType == "dynamic"} {
	puts "Running dynamic analysis..."

	source dynamic.tcl

}

#wipe all;
```

ANEXO M: CODIGO DE CONVERGENCIA PARA PUSHOVER

```
set factor 10;
puts "Trying smaller time step:"
for {set new_i 1} {$new_i <= $factor} {incr new_i 1} {

        set Dincr_new [expr $Dincr/$factor];

        integrator DisplacementControl  $IDctrlNode   $IDctrlDOF $Dincr_new;
        set ok [analyze 1];
        puts "   Convergence loop, step $new_i of $factor: $ok"

        if {$ok != 0} {
                test NormDispIncr   $Tol 2000  0
                algorithm Newton -initial
                set ok [analyze 1 ]
                puts "   Trying Newton with Initial Tangent: $ok"
                test $TestType $Tol $maxNumIter  0
                algorithm $algorithmType
        }

        if {$ok != 0} {
                algorithm Broyden 8
                set ok [analyze 1 ]
                puts "   Trying Broyden: $ok"
                algorithm $algorithmType
        }

        if {$ok != 0} {
                algorithm NewtonLineSearch .8
                set ok [analyze 1 ]
                puts "   Trying NewtonWithLineSearch: $ok"
                algorithm $algorithmType
        }

        if {$ok != 0} {
                algorithm Linear 0.8
                set ok [analyze 1]
                puts "   Trying Linear: $ok"
                algorithm $algorithmType
        }

        if {$ok != 0} {;                                # stop if still fails to converge
                puts "PushOver failed at step $new_i of $factor"
                break;
        };

};
```

```
if {$ok != 0} {;                                # stop if still fails to converge
        puts "PushOver failed at step $i"
        break;
};
```

ANEXO N: CODIGO DE CONVERGENCIA PARA ANÁLISIS DINAMICO

```
# initial parameters: constraints = Transformation
#                     numberer = RCM;
#                     system = BandGeneral;
#                     test = EnergyIncr;
#                     algorithm = Newton;
#                     integrator = Newmark 0.5 0.25;

algorithm KrylovNewton;
set ok [analyze 1 $DtAnalysis]
puts "Trying KrylovNewton: $ok";
algorithm $algorithmType;

if {$ok != 0} {
        algorithm NewtonLineSearch;
        set ok [analyze 1 $DtAnalysis]
        puts "Trying Newton with line search: $ok";
        algorithm $algorithmType;
};

if {$ok != 0} {
        algorithm BFGS;
        set ok [analyze 1 $DtAnalysis]
        puts "Trying BFGS: $ok";
        algorithm $algorithmType;
};

if {$ok != 0} {
        algorithm Broyden 8
        set ok [analyze 1 $DtAnalysis]
        puts "Trying Broyden: $ok"
        algorithm $algorithmType
}

if {$ok != 0} {
set time_factor 100;
set DtAnalysis_new [expr $DtAnalysis/$time_factor];
puts "   Trying smaller time step:"
for {set new_i 1} {$new_i <= $time_factor} {incr new_i 1} {

        set ok [analyze 1 $DtAnalysis_new];
        puts "   Convergence loop, step $new_i of $time_factor: $ok"

        if {$ok != 0} {
                algorithm KrylovNewton;
                set ok [analyze 1 $DtAnalysis_new]
                puts "   Trying KrylovNewton: $ok";
```

```
            algorithm $algorithmType;
        };

        if {$ok != 0} {
            algorithm NewtonLineSearch;
            set ok [analyze 1 $DtAnalysis_new]
            puts "   Trying Newton with line search: $ok";
            algorithm $algorithmType;
        };

        if {$ok != 0} {
            algorithm BFGS;
            set ok [analyze 1 $DtAnalysis_new]
            puts "   Trying BFGS: $ok";
            algorithm $algorithmType;
        };

        if {$ok != 0} {
            algorithm Broyden 8
            set ok [analyze 1 $DtAnalysis_new]
            puts "   Trying Broyden: $ok"
            algorithm $algorithmType
        };

        if {$ok != 0} {;                                    # stop if still fails to converge
            puts "Dynamic analysis failed at step $new_i of $time_factor"
            break;
        };

};
};

if {$ok != 0} {;                                    # stop if still fails to converge
        puts "Dynamic analysis at step $i"
        recorder Node -file $dataDir/COLLAPSE$j.out -time -node 117 217 317 417
51 -dof 1 2 3 reaction;
        break;
};
```

ANEXO Ñ: CODIGO DE EJECUCIÓN DINÁMICA

La región 5 se debe modificar con las etiquetas de los elementos, los cuales a su vez dependen de los números de pisos. Al igual es necesario cambiar el archivo “veinte.tcl” según el número de pisos

```
# Start ground motion iteration:
set GMinput [open "record.txt" r];
set GMs [split [read $GMinput] "\n"]; # each line contains 4 elements
#set GmsP [lrange $GMs 0 end-1];
close $GMinput;

foreach GMrecord "$GMs" {;

wipe
source veinte.tcl

# display deformed shape:
#set ViewScale 10;                              # amplify display of deformed shape
#DisplayModel2D DeformedShape $ViewScale;    # display deformed shape, the scaling factor needs to be adjusted for each model

# Rayleigh Damping

        # calculate damping parameters
        set zeta 0.0250;                                    # percentage of critical damping
        set n 10;
        set a0 [expr $zeta*2.0*$w1*$w3/($w1 + $w3)];    # mass damping coefficient based on first and second modes
        set a1 [expr $zeta*2.0/($w1 + $w3)];                # stiffness damping coefficient based on first and second modes
        set a1_mod [expr $a1*(1.0+$n)/$n];          # modified stiffness damping coefficient used for n modified elements. See Zareian & Medina 2010.

        # assign damping to frame beams and columns
        # command: region $regionID -eleRange $elementIDfirst $elementIDlast rayleigh $alpha_mass $alpha_currentStiff $alpha_initialStiff $alpha_committedStiff
        region 5 -ele  111     121     131     141     112     122     132     142
    11391  11392  12391  12392  13391  13392  14391  14392  114     124     134
    144     11591  11592  12591  12592  13591  13592  14591  14592  116     126
    136     146     11791  11792  12791  12792  13791  13792  14791  14792  118
    128     138     148 11391     11992  12991  12992  13991  13992  14991
    14992 1110    1210    1310    1410 111191  111192          121191
```

```
        121192          131191          131192          141191          141192 1112   1212
        1312    1412 111391   111392          121391          121392          131391
        131392          141391          141392 1114   1214    1314    1414 111591
        111592          121591          121592          131591          131592
        141591          141592 1116   1216    1316    1416 111791   111792
        121791          121792          131791          131792          141791
        141792 1118   1218    1318    1418 111991   111992          121991
        121992          131991          131992          141991          141992 1120   1220
        1320    1420

                rayleigh $a0 0.0 0.0 0.0;                                       # assign
mass proportional damping to structure (only assigns to nodes with mass)


        # define ground motion parameters
                set patternID 1;                # load pattern ID
                set GMdirection 1;              # ground motion direction (1 = x)

                set j [lindex [split $GMrecord] 0];
                set name [lindex [split $GMrecord] 1];
                set NPTS [lindex [split $GMrecord] 2];
                set deltat [lindex [split $GMrecord] 3];
                set scalefactor [lindex [split $GMrecord] 4];
                set GMtime [expr $NPTS*$deltat + 0.00];   # total time of ground
motion + 10 sec of free vibration

        # define the acceleration series for the ground motion
        # syntax:  "Series -dt $timestep_of_record -filePath
$filename_with_acc_history -factor $scale_record_by_this_amount
                set accelSeries "Series -dt $deltat -filePath $name -factor [expr
$scalefactor*$g]";

        # create load pattern: apply acceleration to all fixed nodes with
UniformExcitation
                # command: pattern UniformExcitation $patternID $GMdir -accel
$timeSeriesID
                pattern UniformExcitation $j $GMdirection -accel $accelSeries;

                set dataDir Results
                file mkdir $dataDir


                #recorder Node -file $dataDir/R$j.out -time -node 11 21 31 41 51 -dof
1 2 3 reaction;
                recorder Drift -file $dataDir/Drift$j.out -time -iNode 11 12005 13005
14005 15005 16005 17005 18005 19005 110005 111005 112005 113005 114005
115005 116005 117005 118005 119005 120005 -jNode 12005 13005 14005 15005
```

```
16005 17005 18005 19005 110005 111005 112005 113005 114005 115005 116005
117005 118005 119005 120005 121005 -dof 1 -perpDirn 2;
		recorder Node -file $dataDir/Accel$j.out -time -node 12005 13005
14005 15005 16005 17005 18005 19005 110005 111005 112005 113005 114005
115005 116005 117005 118005 119005 120005 121005 -dof 1 accel;
		recorder Element -file $dataDir/ForceC1F1S1inf$j.out -time -ele 11112
section 1 force;
		recorder Element -file $dataDir/ForceC1F2S1inf$j.out -time -ele 11122
section 1 force;
		recorder Element -file $dataDir/ForceC1F3S1inf$j.out -time -ele 11132
section 1 force;
		recorder Element -file $dataDir/ForceC2F1S1inf$j.out -time -ele 12112
section 1 force;
		recorder Element -file $dataDir/ForceC2F2S1inf$j.out -time -ele 12122
section 1 force;
		recorder Element -file $dataDir/ForceC2F3S1inf$j.out -time -ele 12132
section 1 force;
		recorder Element -file $dataDir/ForceC3F1S1inf$j.out -time -ele 13112
section 1 force;
		recorder Element -file $dataDir/ForceC3F2S1inf$j.out -time -ele 13122
section 1 force;
		recorder Element -file $dataDir/ForceC3F3S1inf$j.out -time -ele 13132
section 1 force;
		recorder Element -file $dataDir/ForceC4F1S1inf$j.out -time -ele 14112
section 1 force;
		recorder Element -file $dataDir/ForceC4F2S1inf$j.out -time -ele 14122
section 1 force;
		recorder Element -file $dataDir/ForceC4F3S1inf$j.out -time -ele 14132
section 1 force;

		recorder Element -file $dataDir/ForceC1F1S2inf$j.out -time -ele 11112
section 2 force;
		recorder Element -file $dataDir/ForceC1F2S2inf$j.out -time -ele 11122
section 2 force;
		recorder Element -file $dataDir/ForceC1F3S2inf$j.out -time -ele 11132
section 2 force;
		recorder Element -file $dataDir/ForceC2F1S2inf$j.out -time -ele 12112
section 2 force;
		recorder Element -file $dataDir/ForceC2F2S2inf$j.out -time -ele 12122
section 2 force;
		recorder Element -file $dataDir/ForceC2F3S2inf$j.out -time -ele 12132
section 2 force;
		recorder Element -file $dataDir/ForceC3F1S2inf$j.out -time -ele 13112
section 2 force;
		recorder Element -file $dataDir/ForceC3F2S2inf$j.out -time -ele 13122
section 2 force;
		recorder Element -file $dataDir/ForceC3F3S2inf$j.out -time -ele 13132
section 2 force;
```

```
		recorder Element -file $dataDir/ForceC4F1S2inf$j.out -time -ele 14112
section 2 force;
		recorder Element -file $dataDir/ForceC4F2S2inf$j.out -time -ele 14122
section 2 force;
		recorder Element -file $dataDir/ForceC4F3S2inf$j.out -time -ele 14132
section 2 force;



		recorder Element -file $dataDir/ForceC1F1S3inf$j.out -time -ele 11112
section $npC force;
		recorder Element -file $dataDir/ForceC1F2S3inf$j.out -time -ele 11122
section $npC force;
		recorder Element -file $dataDir/ForceC1F3S3inf$j.out -time -ele 11132
section $npC force;
		recorder Element -file $dataDir/ForceC2F1S3inf$j.out -time -ele 12112
section $npC force;
		recorder Element -file $dataDir/ForceC2F2S3inf$j.out -time -ele 12122
section $npC force;
		recorder Element -file $dataDir/ForceC2F3S3inf$j.out -time -ele 12132
section $npC force;
		recorder Element -file $dataDir/ForceC3F1S3inf$j.out -time -ele 13112
section $npC force;
		recorder Element -file $dataDir/ForceC3F2S3inf$j.out -time -ele 13122
section $npC force;
		recorder Element -file $dataDir/ForceC3F3S3inf$j.out -time -ele 13132
section $npC force;
		recorder Element -file $dataDir/ForceC4F1S3inf$j.out -time -ele 14112
section $npC force;
		recorder Element -file $dataDir/ForceC4F2S3inf$j.out -time -ele 14122
section $npC force;
		recorder Element -file $dataDir/ForceC4F3S3inf$j.out -time -ele 14132
section $npC force;

		recorder Element -file $dataDir/deformationC1F1S1inf$j.out -time -ele
11112 section 1 deformation ;
		recorder Element -file $dataDir/deformationC1F2S1inf$j.out -time -ele
11122 section 1 deformation;
		recorder Element -file $dataDir/deformationC1F3S1inf$j.out -time -ele
11132 section 1 deformation;
		recorder Element -file $dataDir/deformationC2F1S1inf$j.out -time -ele
12112 section 1 deformation;
		recorder Element -file $dataDir/deformationC2F2S1inf$j.out -time -ele
12122 section 1 deformation;
		recorder Element -file $dataDir/deformationC2F3S1inf$j.out -time -ele
12132 section 1 deformation;
		recorder Element -file $dataDir/deformationC3F1S1inf$j.out -time -ele
13112 section 1 deformation;
```

```
		recorder Element -file $dataDir/deformationC3F2S1inf$j.out -time -ele
13122 section 1 deformation;
		recorder Element -file $dataDir/deformationC3F3S1inf$j.out -time -ele
13132 section 1 deformation;
		recorder Element -file $dataDir/deformationC4F1S1inf$j.out -time -ele
14112 section 1 deformation;
		recorder Element -file $dataDir/deformationC4F2S1inf$j.out -time -ele
14122 section 1 deformation;
		recorder Element -file $dataDir/deformationC4F3S1inf$j.out -time -ele
14132 section 1 deformation;

		recorder Element -file $dataDir/deformationC1F1S2inf$j.out -time -ele
11112 section 2 deformation ;
		recorder Element -file $dataDir/deformationC1F2S2inf$j.out -time -ele
11122 section 2 deformation;
		recorder Element -file $dataDir/deformationC1F3S2inf$j.out -time -ele
11132 section 2 deformation;
		recorder Element -file $dataDir/deformationC2F1S2inf$j.out -time -ele
12112 section 2 deformation;
		recorder Element -file $dataDir/deformationC2F2S2inf$j.out -time -ele
12122 section 2 deformation;
		recorder Element -file $dataDir/deformationC2F3S2inf$j.out -time -ele
12132 section 2 deformation;
		recorder Element -file $dataDir/deformationC3F1S2inf$j.out -time -ele
13112 section 2 deformation;
		recorder Element -file $dataDir/deformationC3F2S2inf$j.out -time -ele
13122 section 2 deformation;
		recorder Element -file $dataDir/deformationC3F3S2inf$j.out -time -ele
13132 section 2 deformation;
		recorder Element -file $dataDir/deformationC4F1S2inf$j.out -time -ele
14112 section 2 deformation;
		recorder Element -file $dataDir/deformationC4F2S2inf$j.out -time -ele
14122 section 2 deformation;
		recorder Element -file $dataDir/deformationC4F3S2inf$j.out -time -ele
14132 section 2 deformation;



		recorder Element -file $dataDir/deformationC1F1S3inf$j.out -time -ele
11112 section $npC deformation;
		recorder Element -file $dataDir/deformationC1F2S3inf$j.out -time -ele
11122 section $npC deformation;
		recorder Element -file $dataDir/deformationC1F3S3inf$j.out -time -ele
11132 section $npC deformation;
		recorder Element -file $dataDir/deformationC2F1S3inf$j.out -time -ele
12112 section $npC deformation;
		recorder Element -file $dataDir/deformationC2F2S3inf$j.out -time -ele
12122 section $npC deformation;
```

```
		recorder Element -file $dataDir/deformationC2F3S3inf$j.out -time -ele
12132 section $npC deformation;
		recorder Element -file $dataDir/deformationC3F1S3inf$j.out -time -ele
13112 section $npC deformation;
		recorder Element -file $dataDir/deformationC3F2S3inf$j.out -time -ele
13122 section $npC deformation;
		recorder Element -file $dataDir/deformationC3F3S3inf$j.out -time -ele
13132 section $npC deformation;
		recorder Element -file $dataDir/deformationC4F1S3inf$j.out -time -ele
14112 section $npC deformation;
		recorder Element -file $dataDir/deformationC4F2S3inf$j.out -time -ele
14122 section $npC deformation;
		recorder Element -file $dataDir/deformationC4F3S3inf$j.out -time -ele
14132 section $npC deformation;


		#recorder Element -file $dataDir/MRFbase-Connec-Mom$j.out -time -
ele 1 2 3 4 force;
		#recorder Element -file $dataDir/MRFbase-Connec-Rot$j.out -time -ele
1 2 3 4 deformation;
		#recorder Element -file $dataDir/MRFcolbase-Mom$j.out -time -ele
3111 3211 3311 3411 force;
		#recorder Element -file $dataDir/MRFcolbase-Rot$j.out -time -ele 3111
3211 3311 3411 deformation;
		#recorder Element -file $dataDir/MRcol1-Mom$j.out -time -ele 3112
3212 3312 3412 force;
		#recorder Element -file $dataDir/MRcol1-Rot$j.out -time -ele 3112
3212 3312 3412 deformation;
		#recorder Element -file $dataDir/MRcol2-Mom$j.out -time -ele 3121
3221 3321 3421 force;
		#recorder Element -file $dataDir/MRcol2-Rot$j.out -time -ele 3121
3221 3321 3421 deformation;
		#recorder Element -file $dataDir/MRFbeam2-Mom$j.out -time -ele
4122 4221 4222 4321 4322 4421 force;
		#recorder Element -file $dataDir/MRFbeam2-Rot$j.out -time -ele 4122
4221 4222 4321 4322 4421 deformation;
		#recorder Element -file $dataDir/MRFbeam3-Mom$j.out -time -ele
4132 4231 4232 4331 4332 4431 force;
		#recorder Element -file $dataDir/MRFbeam3-Rot$j.out -time -ele 4132
4231 4232 4331 4332 4431 deformation;



	# define dynamic analysis parameters
		set DtAnalysis 0.001;		# timestep of analysis
		wipeAnalysis;			# destroy all components of the Analysis
object, i.e. any objects created with system, numberer, constraints, integrator,
algorithm, and analysis commands
```

```
		constraints Transformation;	# how it handles boundary conditions
		numberer RCM;			# renumber dof's to minimize band-width (optimization)
		system UmfPack;			# how to store and solve the system of equations in the analysis
		test NormUnbalance 0.01 100;	# type of convergence criteria with tolerance, max iterations
		set algorithmType  Newton; #ModifiedNewton
		algorithm $algorithmType;


		#algorithm Newton;		# use Newton's solution algorithm: updates tangent stiffness at every iteration
		integrator Newmark 0.5 0.25;	# uses Newmark's average acceleration method to compute the time history
		analysis Transient;		# type of analysis: transient or static
		set NumSteps [expr round(($GMtime + 0.0)/$DtAnalysis)];	# number of steps in analysis

	puts "";
		puts "Beginning Ground Motion Analysis";

		for {set i 1} {$i <= $NumSteps} {incr i 1} {
			set ok [analyze 1 $DtAnalysis];			# actually perform analysis -- returns ok=0 if analysis was successful
			puts "GM step $i out of $NumSteps: $ok";
		    if {$ok != 0} {
				source GM_convergence_loop.tcl;
			};
		}

		puts "End of Ground Motion Analysis";

	# output time at end of analysis
		set currentTime [getTime];	# get current analysis time	(after dynamic analysis)
		puts "The current time is: $currentTime";


	}
```

ANEXO O: CODIGO GRAVITACIONAL PARA LA EJECUCIÓN DINAMICA

Este código pertenece a la parte gravitacional del análisis lateral. Depende del número de pisos, y por lo general se lo define con un nombre aludiendo a la cantidad de pisos. “dos, cuatro, ocho, doce, veinte”

```
# Element ID conventions:
#       1xy   = frame columns with RBS springs at both ends
#       2xy   = frame beams with RBS springs at both ends
#       6xy   = trusses linking frame and P-delta column
#       7xy   = P-delta columns
#       2xya  = frame beams between panel zone and RBS spring
#       3xya  = frame column rotational springs
#       4xya  = frame beam rotational springs
#       5xya  = P-delta column rotational springs
#       4xy00 = panel zone rotational springs
#       500xya = panel zone elements
#       where:
#               x = Pier or Bay #
#               y = Floor or Story #
#               a = an integer describing the location relative to beam-column joint (see description where elements and nodes are defined)

#########################################################################################
#       Set Up & Source Definition

#########################################################################################
        wipe all;                               # clear memory of past model definitions
        model BasicBuilder -ndm 2 -ndf 3;   # Define the model builder, ndm = #dimension, ndf = #dofs
        #source DisplayModel2D.tcl;                 # procedure for displaying a 2D perspective of model
        #source DisplayPlane.tcl;              # procedure for displaying a plane in a model

#########################################################################################
#       Define Analysis Type

#########################################################################################

 # Define type of analysis:  "pushover" = pushover;  "dynamic" = dynamic
        set analysisType "dynamic";
        #set analysisType "pushover"
```

```
	if {$analysisType == "pushover"} {
		set dataDir Concentrated-PanelZone-Pushover-Output;	# name of
output folder
		file mkdir $dataDir;					# create output
folder
	}



#########################################################################
###############################
#	Define Building Geometry, Nodes, Masses, and Constraints

#########################################################################
###############################
	set n 10;

	set NStories 20;				# number of stories
	set NBays 3;				# number of frame bays (excludes bay for
P-delta column)
	set WBay	[expr 20.0*12.0];		# bay width in inches
	set HStory1	[expr 15.0*12.0];		# 1st story height in inches
	set HStoryTyp [expr 13.0*12.0];		# story height of other stories in
inches
	set HBuilding [expr $HStory1 + ($NStories-1)*$HStoryTyp];	# height of
building




# assign boundary condidtions
	source Nodes.tcl
	source Constraints.tcl
	# command:  fix nodeID dxFixity dyFixity rzFixity
	# fixity values: 1 = constrained; 0 = unconstrained
	# fix the base of the building; pin P-delta column at base
	fix 11 1 1 1;
	fix 21 1 1 1;
	fix 31 1 1 1;
	fix 41 1 1 1;
	fix 51 1 1 0;	# P-delta column is pinned


set transfTag 1;
```

```
geomTransf PDelta  $transfTag;       # PDelta transformation


        source Material.tcl

        source CreateWSection.tcl
        source Sections.tcl
        source Elements.tcl
        region 4 -ele  111      121     131     141     112     122     132     142
        11391  11392  12391  12392  13391  13392  14391  14392  114     124     134
        144     11591  11592  12591  12592  13591  13592  14591  14592  116     126
        136     146     11791  11792  12791  12792  13791  13792  14791  14792  118
        128     138     148 11391      11992  12991  12992  13991  13992  14991
        14992 1110    1210    1310    1410 111191  111192          121191
        121192          131191          131192          141191          141192 1112   1212
        1312    1412 111391   111392          121391          121392          131391
        131392          141391          141392 1114   1214    1314    1414 111591
        111592          121591          121592          131591          131592
        141591          141592 1116   1216    1316    1416 111791   111792
        121791          121792          131791          131792          141791
        141792 1118   1218    1318    1418 111991   111992          121991
        121992          131991          131992          141991          141992 1120   1220
        1320    1420

        source ZeroLengthElem.tcl
        source Masses.tcl



#DisplayModel2D NodeNumbers



##########################################################################
#######
#           Gravity Loads & Gravity Analysis                              #
##########################################################################
#######

# apply gravity loads
        #command: pattern PatternType $PatternID TimeSeriesType
        pattern Plain 101 Constant {

                # point loads on leaning column nodes
```

```
# command: load node Fx Fy Mz
set P_PD2 [expr -568.00];     # Floor 2
set P_PD3 [expr -565.20];     # Floor 3
set P_PD21 [expr -511.20];    # Floor 21

load 52 0.0 $P_PD2 0.0;              # Floor 2
load 53 0.0 $P_PD3 0.0;              # Floor 3
load 54 0.0 $P_PD3 0.0;              # Floor 4
load 55 0.0 $P_PD3 0.0;              # Floor 5
load 56 0.0 $P_PD3 0.0;              # Floor 6
load 57 0.0 $P_PD3 0.0;              # Floor 7
load 58 0.0 $P_PD3 0.0;              # Floor 8
load 59 0.0 $P_PD3 0.0;              # Floor 9
load 510 0.0 $P_PD3 0.0;             # Floor 10
load 511 0.0 $P_PD3 0.0;             # Floor 11
load 512 0.0 $P_PD3 0.0;             # Floor 12
load 513 0.0 $P_PD3 0.0;             # Floor 13
load 514 0.0 $P_PD3 0.0;             # Floor 14
load 515 0.0 $P_PD3 0.0;             # Floor 15
load 516 0.0 $P_PD3 0.0;             # Floor 16
load 517 0.0 $P_PD3 0.0;             # Floor 17
load 518 0.0 $P_PD3 0.0;             # Floor 18
load 519 0.0 $P_PD3 0.0;             # Floor 19
load 520 0.0 $P_PD3 0.0;             # Floor 20
load 521 0.0 $P_PD21 0.0;    # Floor 21


# point loads on frame column nodes


set P_F21 [expr -70.21];       # load on each frame node in Floor 2
(exterior)
set P_F22 [expr -46.14];       # load on each frame node in Floor 2
(interior)
set P_F31 [expr -70.21];       # load on each frame node in Floor 3
(exterior)
set P_F32 [expr -46.14];       # load on each frame node in Floor 3
(interior)
set P_F41 [expr -70.21];       # load on each frame node in Floor 4
(exterior)
set P_F42 [expr -46.14];       # load on each frame node in Floor 4
(interior)
set P_F51 [expr -70.21];       # load on each frame node in Floor 5
(exterior)
set P_F52 [expr -46.14];       # load on each frame node in Floor 5
(interior)
```

```
set P_F61 [expr -70.21];      # load on each frame node in Floor 6
(exterior)
set P_F62 [expr -46.14];      # load on each frame node in Floor 6
(interior)
set P_F71 [expr -70.21];      # load on each frame node in Floor 7
(exterior)
set P_F72 [expr -46.14];      # load on each frame node in Floor 7
(interior)
set P_F81 [expr -70.21];      # load on each frame node in Floor 8
(exterior)
set P_F82 [expr -46.14];      # load on each frame node in Floor 8
(interior)
set P_F91 [expr -70.21];      # load on each frame node in Floor 9
(exterior)
set P_F92 [expr -46.14];      # load on each frame node in Floor 9
(interior)
set P_F101 [expr -70.21];     # load on each frame node in Floor 10
(exterior)
set P_F102 [expr -46.14];     # load on each frame node in Floor 10
(interior)
set P_F111 [expr -70.21];     # load on each frame node in Floor 11
(exterior)
set P_F112 [expr -46.14];     # load on each frame node in Floor 11
(interior)
set P_F121 [expr -70.21];     # load on each frame node in Floor 12
(exterior)
set P_F122 [expr -46.14];     # load on each frame node in Floor 12
(interior)
set P_F131 [expr -70.21];     # load on each frame node in Floor 13
(exterior)
set P_F132 [expr -46.14];     # load on each frame node in Floor 13
(interior)
set P_F141 [expr -70.21];     # load on each frame node in Floor 14
(exterior)
set P_F142 [expr -46.14];     # load on each frame node in Floor 14
(interior)
set P_F151 [expr -70.21];     # load on each frame node in Floor 15
(exterior)
set P_F152 [expr -46.14];     # load on each frame node in Floor 15
(interior)
set P_F161 [expr -70.21];     # load on each frame node in Floor 16
(exterior)
set P_F162 [expr -46.14];     # load on each frame node in Floor 16
(interior)
set P_F171 [expr -70.21];     # load on each frame node in Floor 17
(exterior)
set P_F172 [expr -46.14];     # load on each frame node in Floor 17
(interior)
```

```
set P_F181 [expr -70.21];     # load on each frame node in Floor 18
(exterior)
set P_F182 [expr -46.14];     # load on each frame node in Floor 18
(interior)
set P_F191 [expr -70.21];     # load on each frame node in Floor 19
(exterior)
set P_F192 [expr -46.14];     # load on each frame node in Floor 19
(interior)
set P_F201 [expr -70.21];     # load on each frame node in Floor 20
(exterior)
set P_F202 [expr -46.14];     # load on each frame node in Floor 20
(interior)
set P_F211 [expr -61.77];     # load on each frame node in Floor 21
(exterior)
set P_F212 [expr -41.18];     # load on each frame node in Floor 21
(interior)


# Floor 2 loads
load 127 0.0 $P_F21 0.0;
load 227 0.0 $P_F22 0.0;
load 327 0.0 $P_F22 0.0;
load 427 0.0 $P_F21 0.0;

# Floor 3 loads
load 137 0.0 $P_F31 0.0;
load 237 0.0 $P_F32 0.0;
load 337 0.0 $P_F32 0.0;
load 437 0.0 $P_F31 0.0;

# Floor 4 loads
load 147 0.0 $P_F41 0.0;
load 247 0.0 $P_F42 0.0;
load 347 0.0 $P_F42 0.0;
load 447 0.0 $P_F41 0.0;

# Floor 5 loads
load 157 0.0 $P_F51 0.0;
load 257 0.0 $P_F52 0.0;
load 357 0.0 $P_F52 0.0;
load 457 0.0 $P_F51 0.0;

# Floor 6 loads
load 167 0.0 $P_F61 0.0;
load 267 0.0 $P_F62 0.0;
load 367 0.0 $P_F62 0.0;
load 467 0.0 $P_F61 0.0;
```

```
# Floor 7 loads
load 177 0.0 $P_F71 0.0;
load 277 0.0 $P_F72 0.0;
load 377 0.0 $P_F72 0.0;
load 477 0.0 $P_F71 0.0;

# Floor 8 loads
load 187 0.0 $P_F81 0.0;
load 287 0.0 $P_F82 0.0;
load 387 0.0 $P_F82 0.0;
load 487 0.0 $P_F81 0.0;

# Floor 9 loads
load 197 0.0 $P_F91 0.0;
load 297 0.0 $P_F92 0.0;
load 397 0.0 $P_F92 0.0;
load 497 0.0 $P_F91 0.0;

# Floor 10 loads
load 1107 0.0 $P_F101 0.0;
load 2107 0.0 $P_F102 0.0;
load 3107 0.0 $P_F102 0.0;
load 4107 0.0 $P_F101 0.0;

# Floor 11 loads
load 1117 0.0 $P_F111 0.0;
load 2117 0.0 $P_F112 0.0;
load 3117 0.0 $P_F112 0.0;
load 4117 0.0 $P_F111 0.0;

# Floor 12 loads
load 1127 0.0 $P_F121 0.0;
load 2127 0.0 $P_F122 0.0;
load 3127 0.0 $P_F122 0.0;
load 4127 0.0 $P_F121 0.0;

# Floor 13 loads
load 1137 0.0 $P_F131 0.0;
load 2137 0.0 $P_F132 0.0;
load 3137 0.0 $P_F132 0.0;
load 4137 0.0 $P_F131 0.0;

# Floor 14 loads
load 1147 0.0 $P_F141 0.0;
load 2147 0.0 $P_F142 0.0;
load 3147 0.0 $P_F142 0.0;
load 4147 0.0 $P_F141 0.0;
```

```
        # Floor 15 loads
        load 1157 0.0 $P_F151 0.0;
        load 2157 0.0 $P_F152 0.0;
        load 3157 0.0 $P_F152 0.0;
        load 4157 0.0 $P_F151 0.0;

        # Floor 16 loads
        load 1167 0.0 $P_F161 0.0;
        load 2167 0.0 $P_F162 0.0;
        load 3167 0.0 $P_F162 0.0;
        load 4167 0.0 $P_F161 0.0;

        # Floor 17 loads
        load 1177 0.0 $P_F171 0.0;
        load 2177 0.0 $P_F172 0.0;
        load 3177 0.0 $P_F172 0.0;
        load 4177 0.0 $P_F171 0.0;

        # Floor 18 loads
        load 1187 0.0 $P_F181 0.0;
        load 2187 0.0 $P_F182 0.0;
        load 3187 0.0 $P_F182 0.0;
        load 4187 0.0 $P_F181 0.0;

        # Floor 19 loads
        load 1197 0.0 $P_F191 0.0;
        load 2197 0.0 $P_F192 0.0;
        load 3197 0.0 $P_F192 0.0;
        load 4197 0.0 $P_F191 0.0;

        # Floor 20 loads
        load 1207 0.0 $P_F201 0.0;
        load 2207 0.0 $P_F202 0.0;
        load 3207 0.0 $P_F202 0.0;
        load 4207 0.0 $P_F201 0.0;

        # Floor 21 loads
        load 1217 0.0 $P_F211 0.0;
        load 2217 0.0 $P_F212 0.0;
        load 3217 0.0 $P_F212 0.0;
        load 4217 0.0 $P_F211 0.0;


    }

# Gravity-analysis: load-controlled static analysis
```

```
	set Tol 1.0e-6;				# convergence tolerance for test
	constraints Plain;			# how it handles boundary conditions
	numberer RCM;					# renumber dof's to minimize
band-width (optimization)
	system BandGeneral;			# how to store and solve the system of
equations in the analysis (large model: try UmfPack)
	test NormDispIncr $Tol 6;		# determine if convergence has been
achieved at the end of an iteration step
	algorithm Newton;			# use Newton's solution algorithm:
updates tangent stiffness at every iteration
	set NstepGravity 1;			# apply gravity in 10 steps
	set DGravity [expr 1.0/$NstepGravity];	# load increment
	integrator LoadControl $DGravity;	# determine the next time step for an
analysis
	analysis Static;			# define type of analysis: static or
transient
	analyze $NstepGravity;			# apply gravity

	# maintain constant gravity loads and reset time to zero
	loadConst -time 0.0
	puts "Model Built"


##########################################################################
#######
#			Eigenvalue Analysis
##########################################################################
#######

	set pi [expr 2.0*asin(1.0)];		# Definition of pi
	set nEigenI 1;					# mode i = 1
	set nEigenJ 2;					# mode j = 2
	set nEigenK 3;						# mode k = 3
	set nEigenL 4;					# mode l = 4
	set nEigenM 5;						# mode m = 5
	set nEigenO 6;						# mode o = 6
	set nEigenP 7;					# mode p = 7
	set nEigenP1 8;						# mode p1 = 8
	set nEigenP2 9;						# mode p2 = 9
	set nEigenP3 10;				# mode p3 = 10
	set nEigenP4 11;				# mode p4 = 11
	set nEigenP5 12;				# mode p5 = 12
	set nEigenP6 13;				# mode p6 = 13
	set nEigenP7 14;				# mode p7 = 14
	set nEigenP8 15;				# mode p8 = 15
	set nEigenP9 16;				# mode p9 = 16
	set nEigenP10 17;				# mode p10 = 17
	set nEigenP11 18;				# mode p11 = 18
```

```
        set nEigenP12 19;                                   # mode p12 = 19
        set nEigenQ 20;                                     # mode q = 20

        set lambdaN [eigen  -genBandArpack  [expr $nEigenQ]];  # eigenvalue
analysis for nEigenQ modes
        set lambdaI [lindex $lambdaN [expr $nEigenI-1]];  # eigenvalue mode i = 1
        set lambdaJ [lindex $lambdaN [expr $nEigenJ-1]]; # eigenvalue mode j = 2
        set lambdaK [lindex $lambdaN [expr $nEigenK-1]];        # eigenvalue mode
k = 3
        set lambdaL [lindex $lambdaN [expr $nEigenL-1]];        # eigenvalue mode l
= 4
        set lambdaM [lindex $lambdaN [expr $nEigenM-1]];        # eigenvalue mode
m = 5
        set lambdaO [lindex $lambdaN [expr $nEigenO-1]];        # eigenvalue mode
o = 6
        set lambdaP [lindex $lambdaN [expr $nEigenP-1]];# eigenvalue mode p = 7
        set lambdaP1 [lindex $lambdaN [expr $nEigenP1-1]];      # eigenvalue mode
p1 = 8
        set lambdaP2 [lindex $lambdaN [expr $nEigenP2-1]];      # eigenvalue mode
p2 = 9
        set lambdaP3 [lindex $lambdaN [expr $nEigenP3-1]];      # eigenvalue mode
p3 = 10
        set lambdaP4 [lindex $lambdaN [expr $nEigenP4-1]];      # eigenvalue mode
p4 = 11
        set lambdaP5 [lindex $lambdaN [expr $nEigenP5-1]];      # eigenvalue mode
p5 = 12
        set lambdaP6 [lindex $lambdaN [expr $nEigenP6-1]];      # eigenvalue mode
p6 = 13
        set lambdaP7 [lindex $lambdaN [expr $nEigenP7-1]];      # eigenvalue mode
p7 = 14
        set lambdaP8 [lindex $lambdaN [expr $nEigenP8-1]];      # eigenvalue mode
p8 = 15
        set lambdaP9 [lindex $lambdaN [expr $nEigenP9-1]];      # eigenvalue mode
p9 = 16
        set lambdaP10 [lindex $lambdaN [expr $nEigenP10-1]];   # eigenvalue mode
p10 = 17
        set lambdaP11 [lindex $lambdaN [expr $nEigenP11-1]];   # eigenvalue mode
p11 = 18
        set lambdaP12 [lindex $lambdaN [expr $nEigenP12-1]];   # eigenvalue mode
p12 = 19
        set lambdaQ [lindex $lambdaN [expr $nEigenQ-1]];        # eigenvalue mode
q = 20


        set w1 [expr pow($lambdaI,0.5)];                    # w1 (1st mode circular
frequency)
        set w2 [expr pow($lambdaJ,0.5)];                    # w2 (2nd mode circular
frequency)
```

```
	set w3 [expr pow($lambdaK,0.5)];		# w3 (3rd mode circular
frequency)
	set w4 [expr pow($lambdaL,0.5)];		# w4 (4th mode circular
frequency)
	set w5 [expr pow($lambdaM,0.5)];		# w5 (5th mode circular
frequency)
	set w6 [expr pow($lambdaO,0.5)];		# w6 (6th mode circular
frequency)
	set w7 [expr pow($lambdaP,0.5)];		# w7 (7th mode circular
frequency)
	set w8 [expr pow($lambdaP1,0.5)];		# w8 (8th mode circular
frequency)
	set w9 [expr pow($lambdaP2,0.5)];		# w9 (9th mode circular
frequency)
	set w10 [expr pow($lambdaP3,0.5)];		# w10 (10th mode circular
frequency)
	set w11 [expr pow($lambdaP4,0.5)];		# w11 (11th mode circular
frequency)
	set w12 [expr pow($lambdaP5,0.5)];		# w12 (12th mode circular
frequency)
	set w13 [expr pow($lambdaP6,0.5)];		# w13 (13th mode circular
frequency)
	set w14 [expr pow($lambdaP7,0.5)];		# w14 (14th mode circular
frequency)
	set w15 [expr pow($lambdaP8,0.5)];		# w15 (15th mode circular
frequency)
	set w16 [expr pow($lambdaP9,0.5)];		# w16 (16th mode circular
frequency)
	set w17 [expr pow($lambdaP10,0.5)];			# w17 (17th mode
circular frequency)
	set w18 [expr pow($lambdaP11,0.5)];			# w18 (18th mode
circular frequency)
	set w19 [expr pow($lambdaP12,0.5)];			# w19 (19th mode
circular frequency)
	set w20 [expr pow($lambdaQ,0.5)];		# w20 (20th mode circular
frequency)


	set T1 [expr 2.0*$pi/$w1];			# 1st mode period of the
structure
	set T2 [expr 2.0*$pi/$w2];			# 2nd mode period of the
structure
	set T3 [expr 2.0*$pi/$w3];			# 3rd mode period of the
structure
	set T4 [expr 2.0*$pi/$w4];			# 4th mode period of the
structure
	set T5 [expr 2.0*$pi/$w5];			# 5th mode period of the
structure
```

```
        set T6 [expr 2.0*$pi/$w6];                  # 6th mode period of the
structure
        set T7 [expr 2.0*$pi/$w7];                  # 7th mode period of the
structure
        set T8 [expr 2.0*$pi/$w8];                  # 8th mode period of the
structure
        set T9 [expr 2.0*$pi/$w9];                  # 9th mode period of the
structure
        set T10 [expr 2.0*$pi/$w10];                # 10th mode period of the
structure
        set T11 [expr 2.0*$pi/$w11];                # 11th mode period of the
structure
        set T12 [expr 2.0*$pi/$w12];                # 12th mode period of the
structure
        set T13 [expr 2.0*$pi/$w13];                # 13th mode period of the
structure
        set T14 [expr 2.0*$pi/$w14];                # 14th mode period of the
structure
        set T15 [expr 2.0*$pi/$w15];                # 15th mode period of the
structure
        set T16 [expr 2.0*$pi/$w16];                # 16th mode period of the
structure
        set T17 [expr 2.0*$pi/$w17];                # 17th mode period of the
structure
        set T18 [expr 2.0*$pi/$w18];                # 18th mode period of the
structure
        set T19 [expr 2.0*$pi/$w19];                # 19th mode period of the
structure
        set T20 [expr 2.0*$pi/$w20];                # 20th mode period of the
structure


        puts "T1 = $T1 s";                          # display the first mode
period in the command window
        puts "T2 = $T2 s";                          # display the second mode
period in the command window
        puts "T3 = $T3 s";                          # display the third mode
period in the command window
        puts "T4 = $T4 s";                          # display the fourth mode
period in the command window
        puts "T5 = $T5 s";                          # display the fifth mode
period in the command window
        puts "T6 = $T6 s";                          # display the sixth mode
period in the command window
        puts "T7 = $T7 s";                          # display the seventh mode
period in the command window
        puts "T8 = $T8 s";                          # display the eighth mode
period in the command window
```

```
        puts "T9 = $T9 s";                          # display the ninth mode
period in the command window
        puts "T10 = $T10 s";                        # display the tenth mode
period in the command window
        puts "T11 = $T11 s";                        # display the eleventh
mode period in the command window
        puts "T12 = $T12 s";                        # display the twelveth
mode period in the command window
        puts "T13 = $T13 s";                        # display the thirteenth
mode period in the command window
        puts "T14 = $T14 s";                        # display the fourteenth
mode period in the command window
        puts "T15 = $T15 s";                        # display the fifteenth
mode period in the command window
        puts "T16 = $T16 s";                        # display the sixteenth
mode period in the command window
        puts "T17 = $T17 s";                        # display the seventeenth
mode period in the command window
        puts "T18 = $T18 s";                        # display the eighteenth
mode period in the command window
        puts "T19 = $T19 s";                        # display the nineteenth
mode period in the command window
        puts "T20 = $T20 s";                        # display the twentieth
mode period in the command window
```

ANEXO P: CODIGO MATLAB PARA EL POSTPROCESO DE DATOS PUSHOVER

```
close all
clear all
clc

cd ('E:\maestría usfq estructuras\Tesis USFQ\modelos Lentre12\cws 5 10 5\G-
L NP 3\s02\Fiber Hinge and CPH\runner 2\Concentrated-PanelZone-Pushover-
Output')
Data=load('Disp_Roof.out');
m=size(Data);
col=m(1,2);
row=m(1,1);

%Number of floors:
num=2;

%Height of first floor:
H1=15*12;

%Height of typical floor:
Hi=13*12;

%Height of the building:
H=H1+(num-1)*Hi;

%Weight of the building:
W1=800.45;
Wi=796.70;
Wroof=720.38;

W=W1+Wroof+(num-2)*Wi;

for i=1:row
    x(i)=Data(i,2)/H;
    y(i)=Data(i,1)/W;
end

figure('Name','FIGURE 1','NumberTitle','off','Color','white')
plot(x,y)
grid on
set(gca,'GridLineStyle','-','fontsize',25,'XTickLabel',
num2str(get(gca,'XTick').'))
xlabel('Roof Displ. over height (%)','FontName', 'Times New
Roman','FontWeight','bold','fontsize',25);
ylabel('Normalized Vshear/W','FontName', 'Times New
Roman','FontWeight','bold','fontsize',25);
title('PUSHOVER CURVE','FontName', 'Times New
Roman','FontWeight','bold','FontSize',25)

Matriz=[x',y'];
xlswrite('pushover.xlsx',Matriz,'Hoja1','A1');
```