UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Desarrollo de Modelos No Lineales para Pórticos de Acero Resistentes a Momento

Diego Eduardo Torres Viteri

Ingeniería Civil

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero Civil

Quito, 16 de mayo de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Desarrollo de Modelos No Lineales para Pórticos de Acero Resistentes a Momento

Diego Eduardo Torres Viteri

Nombre del profesor, Título académico **Pablo Torres, Ph. D.**

Quito, 16 de mayo de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: Diego Eduardo Torres Viteri

Código: 00208341

Cédula de identidad: 1750275149

Lugar y fecha: Quito, 16 de mayo de 2023

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UNPUBLISHED DOCUMENT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 capstone 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.

RESUMEN

En este trabajo se encuentran directrices generales y pautas específicas para un análisis estructural no lineal de pórticos de acero resistentes a momento. Considerando estas directrices basadas en publicaciones existentes sobre la materia, se desarrolló una aplicación en MATLAB App Designer que ejecuta tres tipos de análisis: análisis lineal gravitacional, análisis no lineal estático y análisis no lineal dinámico. El programa funciona en base a subrutinas desarrolladas en MATLAB que generan archivos Tool Comand Language (Tcl) para el software de uso libre Open System for Earthquake Engineering Simulation (OpenSees). Este proyecto va a servir para evaluar el desempeño de edificios existentes y diseñar estructuras de acero en países sísmicos.

Palabras clave: MATLAB, Tcl, OpenSees, análisis no lineal, pórticos de acero y App Designer.

ABSTRACT

In this work, general guidelines and specific directions for a nonlinear structural analysis of steel moment-resisting frames are found. Considering these guidelines based on existing publications on the subject, an application was developed in MATLAB App Designer that performs three types of analysis: linear gravity analysis, nonlinear static analysis, and nonlinear dynamic analysis. The program works on the basis of subroutines developed in MATLAB that generate Tool Command Language (Tcl) files for the open-source software Open System for Earthquake Engineering Simulation (OpenSees). This project will be used to evaluate the performance of existing buildings and design steel structures in seismic countries.

Key words: MATLAB, Tcl, OpenSees, nonlinear analysis, steel moment frames, and App Designer.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Este proyecto consiste en el desarrollo de un código en MATLAB que genera archivos Tcl para OpenSees con el propósito de llevar a cabo análisis no lineales de pórticos de acero resistentes a momento, todo esto a través de una aplicación creada en App Designer. A continuación, se encuentran algunas definiciones de términos muy importantes para comprender mejor este tema.

Se sabe que MATLAB es un paquete de software para cálculos en ingeniería, ciencia y matemáticas aplicadas. Ofrece un poderoso lenguaje de programación, excelentes gráficos y una amplia gama de conocimientos especializados. MATLAB fue desarrollado por la marca registrada The MathWorks, Inc (*matlab_adv.pdf*, s/f, p. 3).

Por otro lado, Tcl es un lenguaje de programación dinámico con gran potencial, adecuado para una extensa variedad de usos, incluyendo aplicaciones web y de escritorio, conexión en red, administración, pruebas, entre otras. Además de ser una fuente abierta y fácil de usar en las empresas, Tcl es un lenguaje bastante desarrollado que sigue en evolución y sirve para varias plataformas. Destacando su fácil instalación y con la posibilidad de extenderse (*Tcl Developer Site*, s/f).

Mientras tanto, OpenSees es un software de creación de aplicaciones para simular el rendimiento de sistemas estructurales y geotécnicos sometidos a terremotos. El objetivo del desarrollo de OpenSees es mejorar la modelación y las simulaciones computacionales en ingeniería sísmica mediante la programación de código abierto (*Open System for Earthquake Engineering Simulation - Home Page*, s/f).

Por su parte, MATLAB App Designer es un medio de generación interactivo para crear una aplicación y configurar su comportamiento. Facilita una versión incorporada del editor de

MATLAB y un amplio grupo de componentes interactivos de la interfaz del usuario (IU). Asimismo, cuenta con un administrador de diseño de cuadrículas que es de gran utilidad para ordenar la IU. Además, permite compartir aplicaciones empaquetándolas en los archivos del instalador mediante el menú de herramientas de App Designer o desarrollando una app de escritorio independiente (*Desarrollar apps mediante App Designer - MATLAB & Simulink - MathWorks América Latina*, s/f).

Sumado a los programas mencionados anteriormente, el análisis no lineal brinda las herramientas para calcular la respuesta estructural más allá del rango elástico, incluyendo los deterioros tanto de resistencia como rigidez asociados con el comportamiento inelástico del material y grandes desplazamientos (NIST, 2010, p. 1). Aparte de eso, las aplicaciones del análisis estructural no lineal para el diseño sísmico de edificios se han vuelto cada vez más frecuentes en los últimos años para evaluar el desempeño de las estructuras. En este tipo de procedimiento se destacan el análisis no lineal estático (pushover) y el análisis no lineal dinámico (response history).

Las medidas típicas que son calculadas de la respuesta estructural en un análisis no lineal se conocen como “parámetros de demanda”, estos parámetros engloban: derivas y aceleraciones de cada piso; deformaciones por fluencia o elementos con “deformación controlada”; y demandas de fuerzas en los componentes con “fuerza controlada” que se espera se mantengan elásticos.

Los parámetros de demanda calculados se utilizan para: evaluar la conformidad con los criterios de aceptación mediante procedimientos prescriptivos basados en códigos; predecir métricas de desempeño explícitas relacionadas con funcionalidad, pérdidas y seguridad; o cuantificar el riesgo de colapso de una estructura (Applied Technology Council, 2017, p. 1–2).

Por último, este tema es de gran relevancia para el área de Ingeniería Civil y el contexto ecuatoriano. Se tiene conocimiento de que Ecuador es un país altamente sísmico ya que se ubica en el Cinturón de Fuego del Pacífico. En esta zona, la placa tectónica de Nazca se subduce bajo la placa Sudamericana, provocando terremotos de magnitudes de hasta 8.8 en la escala de Richter. Por tanto, las estructuras siempre van a estar sometidas a eventos sísmicos. Debido a todos estos factores, el desarrollo de modelos no lineales para pórticos de acero resistentes a momento va a permitir llevar a cabo excelentes diseños y evaluar el desempeño de edificios ante cargas extremas.

En la próxima sección se van a explicar las características generales del análisis no lineal; los detalles específicos que conllevan un análisis estructural no lineal para pórticos de acero resistentes a momento; los códigos desarrollados en MATLAB para el modelo de cálculo; la generación de los archivos Tcl para OpenSees; y la aplicación creada en App Designer.

DESARROLLO DEL TEMA

Directrices generales para un análisis estructural no lineal

Visión general del procedimiento de modelización y análisis no lineal.

Evaluación y diseño sísmico basado en desempeño.

El diseño basado en desempeño es un procedimiento formal para diseñar edificios, el cual toma en cuenta objetivos establecidos de rendimiento estructural ante futuros terremotos. El análisis estructural no lineal es una etapa muy importante en el proceso de evaluación y diseño sísmico basado en desempeño (Applied Technology Council, 2017, p. 2–1). A continuación, se presentan los pasos necesarios para llevar a cabo un diseño basado en desempeño:

1. Definir el propósito y objetivos del análisis no lineal.
2. Entender el comportamiento estructural y los modos de falla.
3. Definir los parámetros de demanda y los criterios de aceptación.
4. Desarrollar el modelo analítico estructural.
5. Realizar el análisis.
6. Evaluar el desempeño sísmico del edificio a través de los criterios de aceptación.
7. Revisión por pares y documentación.

Parámetros de demanda.

Las decisiones en el modelo no lineal están muy influenciadas por el tipo de los parámetros de demanda que necesitan ser evaluados en el análisis estructural (Applied Technology Council, 2017, p. 2–3). Los valores de respuesta deseados son los siguientes:

- Deriva media máxima de los pisos.
- Deriva máxima de piso.

- Deriva residual de piso.
- Aceleración media máxima de los pisos.
- Deformaciones.
- Fuerzas.
- Probabilidad de Colapso.

Visión general del enfoque de modelización no lineal.

Una vez que los objetivos del análisis se establecen y los parámetros de demanda fueron identificados, lo siguiente para crear el modelo estructural no lineal es decidir que componentes del edificio se van a incluir y como se los va a idealizar dentro del modelo (Applied Technology Council, 2017, p. 2–7).

Clasificación de componentes por su comportamiento esperado.

Después de reconocer los componentes del modelo estructural, el siguiente paso es clasificar cada componente como controlado por deformación o controlado por fuerza. Los elementos controlados por deformación son aquellos que tienen capacidad de deformación inelástica fiable y un decaimiento gradual de su resistencia. Por otro lado, las acciones controladas por fuerza están asociadas a los elementos que se desea se mantengan en el rango elástico (Applied Technology Council, 2017, p. 2–8).

Componentes controlados por deformación.

Las acciones de los elementos controlados por deformación deben ser modeladas de tal forma que capturen su respuesta no lineal esperada en un evento sísmico. Para esto, es necesario considerar las degradaciones de resistencia y rigidez. La Figura 1 (Suzuki y Lignos, 2015) muestra una ilustración de datos de prueba pertenecientes a tres columnas de acero idénticas que fueron sometidas a tres protocolos de carga lateral distintos. La curva azul se obtiene al

aplicar una carga en un solo sentido (monótona) a la columna, en cambio, la curva roja se genera al aplicar una carga en ambos sentidos (cíclica) a la columna. Por último, la función verde representa como colapsaría la columna durante un terremoto real. Como conclusión, en las tres curvas se evidencia que las degradaciones de resistencia y rigidez evolucionan debido a la carga sísmica (Applied Technology Council, 2017a, pp. 2–8).

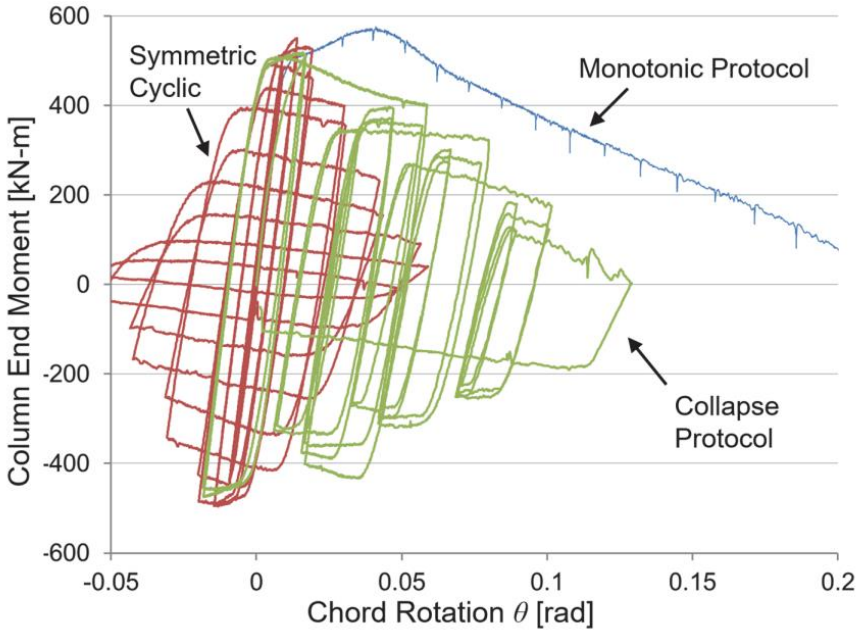

Figura 1. Respuestas de una columna de acero sujeta a tres protocolos de carga distintos.

Además, la Figura 2 (Suzuki y Lignos, 2015) enseña las envolventes monotónica y cíclica para los ensayos de columnas mostrados en la Figura 1. En este caso, la envolvente monotónica se obtiene directamente de la prueba con carga monótona. Mientras que la envolvente cíclica se consigue de los datos del ensayo cíclico, es decir, esta curva se forma conectando los puntos máximos de carga para cada nivel de deformación (PEER/ATC, 2010), por tanto, depende del protocolo de carga que se aplicó en el ensayo.

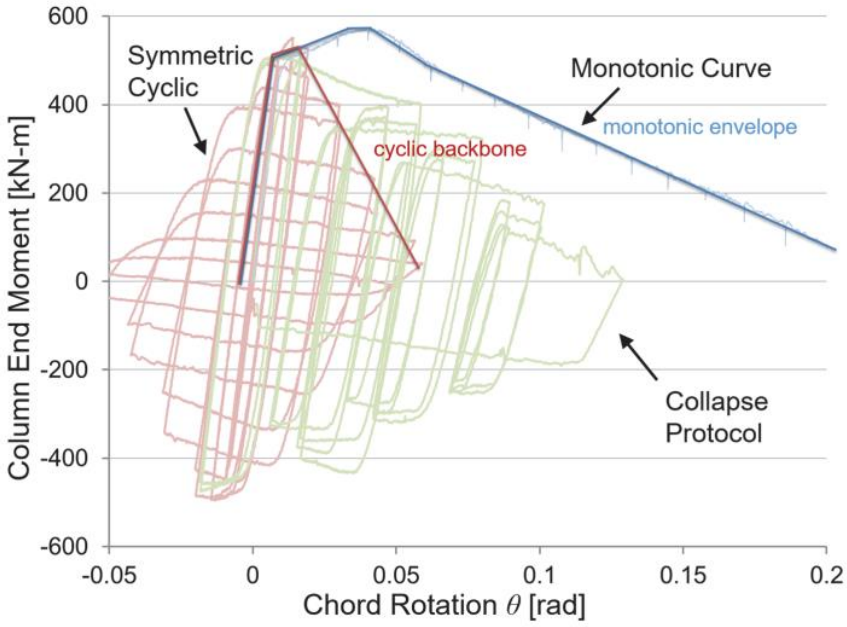

Figura 2. Envolvente monotónica y curva backbone cíclica superpuestas a los datos de los ensayos de columnas de acero.

Puede que lo más importante en un análisis no lineal sea el enfoque recomendado para el modelado de componentes. En este proyecto, o en cualquier modelo no lineal, las respuestas estructurales de vigas, columnas y paneles zonales se modelaron en base a la envolventes monotónica y cíclica. Como se muestra en la Figura 3 (NIST, 2017), el enfoque se centra en especificar dos curvas generalizadas de fuerza-deformación: la curva envolvente monotónica y la curva backbone cíclica (predegradada). En el análisis no lineal estático los componentes se calibran con la envolvente cíclica y en el análisis no lineal dinámico los modelos se ajustan utilizando la envolvente monotónica (Applied Technology Council, 2017, p. 2–13).

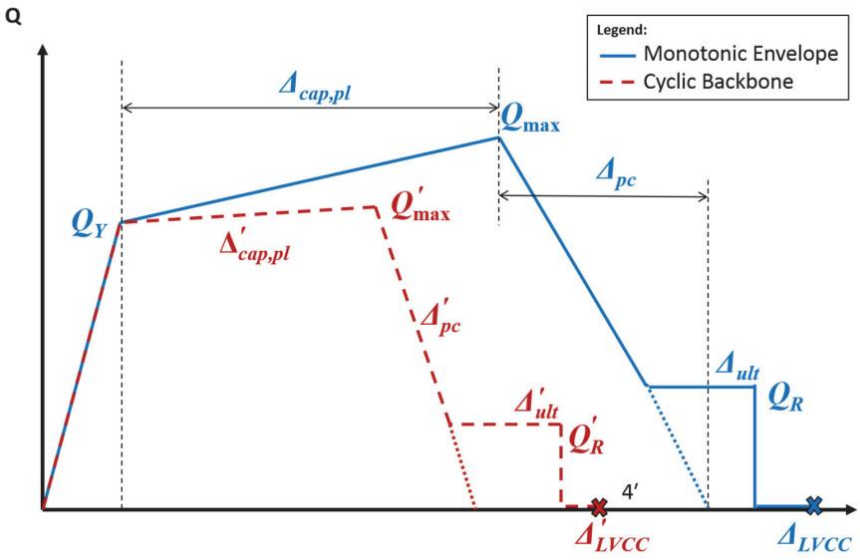

Figura 3. Backbones cíclica y monotónica con puntos de control.

En la figura, se aplica la siguiente notación:

Q_y = resistencia a la fluencia del elemento

Q_{max} = resistencia máxima del elemento, carga monotónica

Q'_{max} = resistencia máxima del elemento, carga cíclica

Q_R = resistencia residual del elemento, carga monotónica

Q'_R = resistencia residual del elemento, carga cíclica

$\Delta_{cap,pl}$= deformación plástica, carga monotónica

$\Delta'_{cap,pl}$= deformación plástica, carga cíclica

Δ_{pc}= deformación efectiva posterior al punto máximo, carga monotónica

Δ'_{pc}= deformación efectiva posterior al punto máximo, carga cíclica

Δ_{ult}= deformación última, carga monotónica

Δ'_{ult}= deformación última, carga cíclica

Δ_{LVCC}= deformación donde se pierde la capacidad de carga, carga monotónica

Δ'_{LVCC} = deformación donde se pierde la capacidad de carga, carga cíclica

Componentes controlados por fuerza.

Las acciones de los elementos categorizadas como controladas por fuerza usualmente se modelan elásticamente, donde las demandas de fuerza resultantes se comparan con los límites de resistencia de los componentes. Para considerar las incertidumbres en los cálculos, se especifican factores de carga que se multiplican con las demandas y factores de resistencia que se aplican a la capacidad nominal o esperada de los elementos (Applied Technology Council, 2017, p. 2–16).

Tipos de modelos de componentes.

Como se ilustra en los elementos viga-columna de la Figura 4 (NIST, 2010), los modelos para el análisis no lineal pueden ir desde modelos uniaxiales de resorte o rótula, pasando por modelos más fundamentales de tipo fibra, hasta modelos detallados de elementos finitos continuos. En general, todos los modelos son fenomenológicos ya que se basan en calibraciones empíricas de comportamientos obtenidos hasta cierto nivel de idealización. No obstante, los modelos de plasticidad concentrada son bastante fenomenológicos porque las funciones que describen su comportamiento estructural están establecidas con calibraciones que toman en cuenta el comportamiento global de los elementos (Applied Technology Council, 2017, p. 2–17).

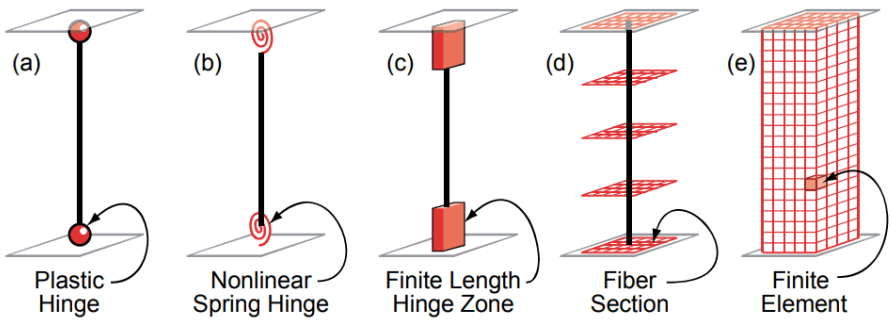

Figura 4. Rangos de los tipos de modelos estructurales.

Los modelos de rótulas y resortes tienen la ventaja de ser eficientes computacionalmente debido a que modelan los efectos no lineales en regiones localizadas de la estructura con pocos grados de libertad. Además, los modelos de resortes concentrados, típicamente son implementados para capturar respuestas de un solo grado de libertad, pero pueden incluir respuestas multiaxiales a través de superficies plastificadas u otros medios (Applied Technology Council, 2017, p. 2–17).

Resumen de los tipos de modelos de elementos.

- **Elementos de resorte:** Los elementos de resorte pueden representar un solo grado de libertad en 2D o 3D, traslación o rotación, o varios grados de libertad acoplados o desacoplados, dependiendo de su aplicación y las capacidades del software. Entre los ejemplos más comunes están los resortes de flexión concentrados, resortes axiales, resortes de paneles zonales, resortes de suelo, o resortes de tensión/compresión. En algunas implementaciones, las propiedades de los resortes se pueden acoplar entre dos o más acciones, como resortes que representan la respuesta axial y de flexión en las rótulas de vigas y columnas (Applied Technology Council, 2017, p. 2–19).
- **Elementos de línea:** Los elementos de línea típicamente se usan para modelar vigas y columnas en pórticos resistentes a momento y pórticos arriostrados; puntales en marcos arriostrados y armaduras; y vigas de acople en sistemas acoplados con muros (Applied Technology Council, 2017, pp. 2–19).

Modelos ilustrativos de pórticos a momento.

El marco de acero resistente a momento, mostrado en la Figura 5 (Applied Technology Council, 2017b, p. 3–2), tiene uniones de secciones de vigas reducidas (RBS) totalmente restringidas, donde las regiones de las rótulas están desplazadas de las caras de las columnas. En este caso, los modelos de vigas y columnas consisten en elementos de línea elásticos con

rótulas concentradas en sus extremos. Los paneles zonales se muestran representados por elementos offset, esto sirve para establecer su tamaño finito, el cual puede ser modelado con elementos flexibles o rígidos. Adicionalmente, las rótulas de las vigas RBS están alejadas de las caras de las columnas utilizando elementos elásticos de pequeña longitud (Applied Technology Council, 2017a, p. 2–20).

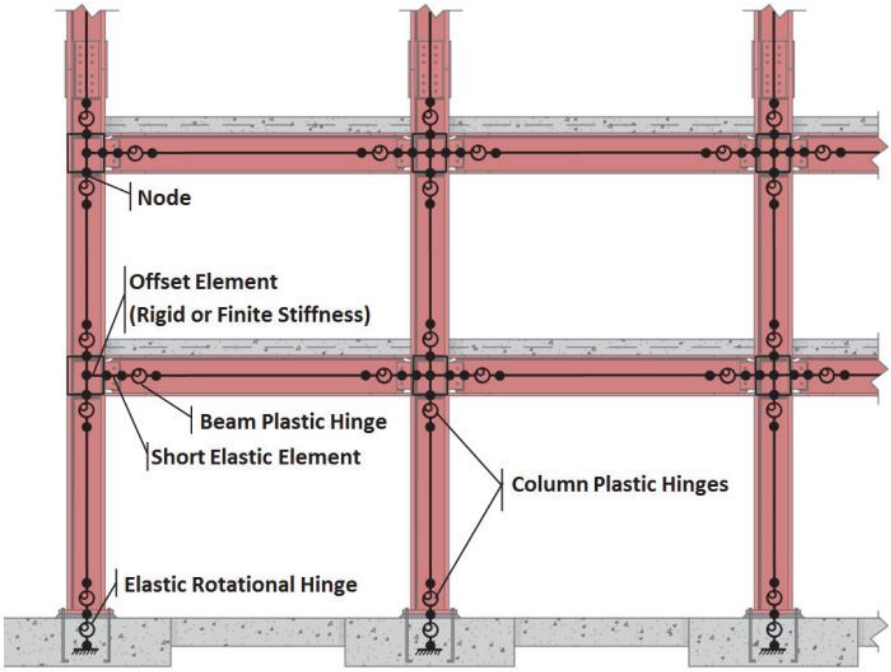

Figura 5. Modelo de análisis para un pórtico de acero resistente a momento.

Requisitos generales para modelos estructurales no lineales.

Guías generales para la modelación.

En general, el objetivo de una evaluación basada en desempeño utilizando un análisis no lineal dinámico es simular la respuesta del edificio lo más real posible para obtener una medida de su rendimiento (Applied Technology Council, 2017a, p. 3–1). La intención es realizar el análisis de forma práctica y transparente para conseguir los resultados que se desean. Para desarrollar modelos realistas hay que tomar en cuenta los siguientes principios:

- Selección de componentes que afectan significativamente la respuesta.

- Geometría y tamaño finito de los elementos.
- Condiciones de borde.
- Apropiada sofisticación del modelo para los niveles de demanda.

Masa sísmica.

Toda la masa que está sujeta físicamente a la estructura debería ser incluida en el modelo de análisis. Asimismo, es recomendable incluir la masa de carga muerta superpuesta y la masa de ciertos tipos de carga viva a la masa sísmica (Applied Technology Council, 2017a, p. 3–2).

Cargas gravitacionales.

Las cargas gravitacionales deberían ser especificadas como las cargas de gravedad esperadas durante un terremoto (un caso de carga único). Todas las cargas gravitacionales deberían ser aplicadas al modelo estructural y después los movimientos sísmicos del suelo tendrían que ser impuestos con las cargas verticales manteniendo su posición en la estructura (Applied Technology Council, 2017a, p. 3–3).

No linealidades geométricas.

Aquí se consideran los efectos P-delta, definidos como las fuerzas (o momentos) desestabilizadoras ejercidas por las cargas gravitacionales en los miembros portantes de carga vertical debido a desplazamientos laterales, estas fuerzas pueden tener un efecto significativo en las rigidez lateral, deformaciones, y estabilidad de una estructura (Applied Technology Council, 2017a, p. 3–3).

Propiedades del material.

Es necesario basar los parámetros y criterios del modelo en propiedades esperadas de los materiales y componentes. En estructuras de acero se considera un factor de esfuerzo de

fluencia probable (Ry) para determinar la rigidez esperada, resistencia prevista, y respuesta inelástica de los elementos estructurales (Applied Technology Council, 2017a, p. 3–5).

En ausencia de datos específicos para un análisis, las propiedades esperadas para modelar miembros de acero están dadas en la Tabla 1, la cual se basó en criterios publicados en ANSI/AISC 341-05, *Seismic Provisions for Structural Steel Buildings* (AISC, 2005); ACI 318-14, *Building Code Requirements for Structural Concrete and Commentary* (ACI, 2014); ASCE/SEI 41-13, *Seismic Evaluation and Retrofit of Existing Buildings* (ASCE, 2014); *An Alternative Procedure for Seismic Analysis and Design of Tall Buildings Located in the Los Angeles Region* (LATBSDC, 2015); y *Guidelines for Performance-Based Seismic Design of Tall Buildings* (PEER, 2017).

Tabla 1. Resistencia esperada del acero.

Material	Expected Strength
Structural Steel – Hot Rolled Structural Shapes and Bars	
ASTM A36/A36M	$1.5F_y$
ASTM A1043/1043M Gr. 36 (250)	$1.3F_y$
ASTM A992/A992M	$1.1F_y$
ASTM A572/572M Gr. 50 (345) or 55 (380)	$1.1F_y$
ASTM A913/A913M Gr. 50 (345), 60 (415), 65 (450), or 70 (485)	$1.1F_y$
Structural Steel – Hollow Structural Sections (HSS)	
ASTM A500/A500M Gr. B	$1.4F_y$
ASTM A500/A500M Gr. C	$1.3F_y$
ASTM A53/A53M	$1.6F_y$
ASTM A1085/A1085M	$1.2F_y$
Structural Steel – Plates, Strips, and Sheets	
ASTM A36/A36M	$1.3F_y$
ASTM A572/A572M Gr. 42 (290)	$1.3F_y$
ASTM A572/A572M Gr. 50 (345), Gr. 55 (380)	$1.1F_y$
ASTM A588/A588M	$1.1F_y$
Reinforcing Steel	
ASTM A615/A615M Gr. 60 (420)	$1.2F_y$
ASTM A615/A615M Gr. 75 (520) and Gr. 80 (550)	$1.1F_y$
ASTM A706/A706M Gr. 60 (420) and Gr. 80 (550)	$1.2F_y$

Modelación del amortiguamiento.

El amortiguamiento viscoso (fuerza dependiente de la velocidad) se usa para representar la disipación de energía que no se modela de otro modo en los componentes de la estructura. Matemáticamente, los efectos del amortiguamiento viscoso son las fuerzas asociadas con el término proporcional a la velocidad [C]{v} en la ecuación que gobierna el movimiento del sistema y sus elementos (Applied Technology Council, 2017a, p. 3–14). Cabe mencionar que el amortiguamiento histerético, proveniente de regiones estructurales que responden inelásticamente, contribuye a la mayor disipación de energía en el sistema (Applied Technology Council, 2017a, p. 3–15).

Tratamiento del sistema gravitacional.

Como mínimo, en el modelo de análisis no lineal se deben considerar los efectos P-delta provenientes de las fuerzas del sistema a cargas gravitacionales. Esto se puede llevar a cabo usando una “columna inclinada”. La columna de la Figura 6 (Applied Technology Council, 2017a, p. 3–26) permite añadir cargas gravitacionales al modelo estructural, pero no toma en cuenta ninguna rigidez lateral o resistencia del sistema gravitacional.

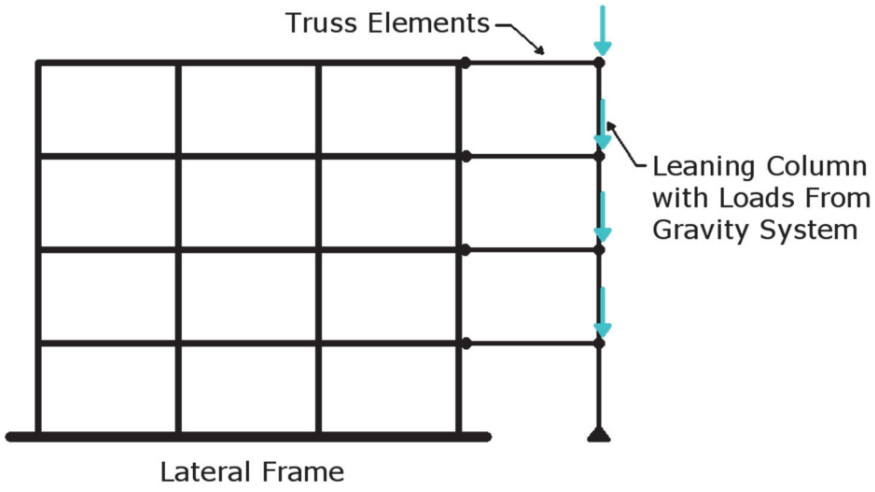

Figura 6. Columna inclinada.

Procedimiento del análisis no lineal estático (pushover).

En el procedimiento no lineal estático, el modelo estructural es sometido a una carga lateral progresiva cuya distribución representa las fuerzas de inercia esperadas durante un terremoto. La carga lateral se aplica hasta que los desplazamientos impuestos alcancen el "desplazamiento objetivo", el cual representa la demanda de desplazamiento que un sismo impondría a la estructura (NIST, 2010, p. 21).

Resumen del procedimiento.

1. Crear y calibrar el modelo de análisis no lineal.
2. Definir y aplicar las fuerzas estáticas gravitacionales.
3. Definir un patrón de carga lateral sísmica equivalente.
4. Establecer un desplazamiento objetivo para una intensidad sísmica específica.
5. Aumentar progresivamente la carga lateral equivalente hasta que se alcance el desplazamiento objetivo.
6. Evaluar las derivas de piso y otros parámetros de demanda (rotaciones de rótulas, fuerzas en los elementos) contra los criterios de aceptación específicos para el desplazamiento objetivo prescrito.

Requisitos para la modelación de componentes.

Los requisitos de modelación de elementos para un análisis no lineal estático son más simples que para un análisis no lineal dinámico ya que los modelos de componentes no necesitan capturar la respuesta cíclica histerética. Para compensar esto, es necesario calibrar los modelos de componentes con plasticidad concentrada utilizando la envolvente cíclica y no la envolvente monotónica (Applied Technology Council, 2017a, p. 4–2).

Carga estática equivalente.

El patrón de carga debe ser tal que la forma desplazada del edificio se aproxime a la respuesta estructural real bajo cargas dinámicas. En la práctica, esto es muy difícil de conseguir, excepto para edificios de baja altura con geometría y sistemas regulares, cuya deformación inelástica es similar al modo fundamental de vibración elástico (Applied Technology Council, 2017a, p. 4–3).

Desplazamiento objetivo.

Existen dos métodos para determinar el desplazamiento objetivo en función de la intensidad sísmica (Sa(T1)) y el periodo fundamental de vibración (T1). Estos son el "método del coeficiente" y el "método de capacidad espectral". El método del coeficiente es más sencillo de implementar, mientras que el método de capacidad espectral tiene la virtud de proveer una interpretación gráfica de la respuesta (Applied Technology Council, 2017a, p. 4–3).

Limitaciones del análisis no lineal estático.

Las principales limitaciones del análisis no lineal estático se derivan de su inherente incapacidad de capturar la respuesta estructural dinámica, incluyendo lo siguiente:

- Comportamiento dinámico de múltiples modos de vibración.
- Degradación cíclica del material y componentes.
- Efectos dependientes de la velocidad.
- Comportamiento de sistemas con amortiguadores sísmicos.
- Características únicas de los movimientos sísmicos del suelo.

Procedimiento del análisis no lineal dinámico (response history).

El modelo de análisis no lineal dinámico incorpora el comportamiento inelástico de los elementos bajo movimientos sísmicos cíclicos. Además, el procedimiento no lineal dinámico

simula explícitamente la disipación de energía histerética en el rango no lineal. La respuesta dinámica es calculada para movimientos sísmicos de entrada, obteniendo datos históricos de respuesta en los parámetros de demanda (NIST, 2010, p. 23). En otras palabras, en el análisis no lineal dinámico el modelo estructural es sometido a uno o más registros sísmicos que son seleccionados y escalados de acuerdo a los objetivos del análisis (Applied Technology Council, 2017a, p. 5–1).

Caracterización de los movimientos sísmicos.

El análisis no lineal response history ofrece una representación más precisa de la respuesta temporal de la estructura a los movimientos sísmicos de entrada. El objetivo es predecir una respuesta estructural media para un nivel específico de demanda sísmica (Applied Technology Council, 2017a, p. 5–1). Para caracterizar los movimientos del terreno hay que tomar las siguientes decisiones:

- Tipo de método de evaluación.
- Espectros objetivo.
- Número de movimientos sísmicos.
- Componentes de los movimientos sísmicos.
- Selección de movimientos sísmicos.
- Medición de la intensidad del movimiento sísmico.
- Escalamiento de los movimientos sísmicos.
- Adaptación espectral.
- Aplicación de los movimientos al modelo estructural.
- Modificaciones de los efectos de la interacción suelo-estructura.

Consideraciones de la modelación numérica.

El análisis no lineal dinámico requiere discretización en el dominio del tiempo (mediante la modelación de la duración del movimiento sísmico y la respuesta estructural asociada dividiendo el análisis en una serie de pasos temporales discretos, ejecutados secuencialmente), así como en el dominio espacial (Applied Technology Council, 2017a, p. 5–4). En cuanto a los errores temporales, estos se acumulan en gran medida cuando se utilizan pasos en el tiempo muy largos.

Es común llevar a cabo iteraciones no lineales dentro de cada paso temporal en un análisis no lineal. Cada paso en el tiempo debería converger dentro de un pequeño número de iteraciones (1-20) si el tamaño del paso temporal se escoge correctamente. No obstante, es usual encontrar problemas de convergencia mientras se ejecuta un análisis no lineal dinámico, esto suele requerir múltiples iteraciones de revisión y modificación del modelo para lograr finalmente la convergencia (Applied Technology Council, 2017a, p. 5–7).

Completar el análisis e interpretar los resultados de la respuesta estructural.

Después de construir el modelo y realizar comprobaciones de control de calidad, el análisis no lineal dinámico puede ejecutarse bajo el conjunto de registros de movimientos del suelo. Finalmente, a pesar de que los movimientos del terreno son seleccionados o espectralmente adaptados al espectro de respuesta objetivo, la respuesta estructural debido a los diferentes movimientos sísmicos típicamente exhibe una variabilidad considerable (Applied Technology Council, 2017a, p. 5–9).

Directrices específicas para un análisis no lineal de pórticos de acero resistentes a momento

Modelos de componentes de plasticidad concentrada.

Para calcular los momentos ($kip \cdot in$) y rotaciones (rad) de las envolventes cíclicas para las rótulas plásticas de vigas y columnas se utilizaron las ecuaciones del documento *Guidelines for Nonlinear Structural Analysis for Design of Buildings, Part IIa – Steel Moment Frames* (Applied Technology Council, 2017b).

Modelo de vigas RBS.

Resistencia efectiva a la fluencia (My).

$$M_y = \beta M_{p,exp} = \beta Z R_y F_y$$

$\beta = factor\ de\ incremento = 1.1\ para\ conexiones\ de\ vigas\ RBS$

$M_{p,exp} = momento\ plástico\ esperado$

$Z_{RBS} = módulo\ de\ sección\ plástico\ de\ la\ sección\ reducida$

$R_y = factor\ de\ esfuerzo\ de\ fluencia\ probable$

$F_y = esfuerzo\ de\ fluencia\ del\ acero$

Rigidez elástica (Ke).

$$K_e = \alpha_e EI/L$$

$\alpha_e = coeficiente\ de\ rigidez = 60$

$EI = rigidez\ de\ la\ sección\ transversal$

$L = longitud\ de\ la\ viga$

Es necesario ajustar la rigidez de la viga conectada a las rótulas plásticas.

$$EI^* = EI/(1-6/\alpha_e)$$

$EI^* = rigidez\ efectiva\ de\ la\ viga$

Momento máximo (Mu).*

$$M_u^* = 1.15M_y$$

Rotación pre-pico (θp).*

$$\theta_p^* = 0.55\left(\frac{h}{t_w}\right)^{-0.5}\left(\frac{b_f}{2t_f}\right)^{-0.7}\left(\frac{L_b}{r_y}\right)^{-0.5}\left(\frac{L}{d}\right)^{0.8}$$

$h/t_w = relación\ profundidad/espesor\ del\ alma$

$b_f/2t_f = relación\ ancho/espesor\ del\ ala$

$L_b/r_y = longitud\ no\ arriostrada\ dividida\ para\ el\ radio\ de\ giro$

$L/d = relación\ luz/profundidad\ de\ la\ viga$

Rotación post-pico (θpc).*

$$\theta_{pc}^* = 20.0\left(\frac{h}{t_w}\right)^{-0.8}\left(\frac{b_f}{2t_f}\right)^{-0.1}\left(\frac{L_b}{r_y}\right)^{-0.6}$$

Resistencia residual (Mr).*

$$M_r^* = 0.3\ M_y$$

Rotación última (θult).*

$$\theta_{ult}^{*} = 0.08$$

Modelo de columnas de ala ancha.

Momento efectivo a la fluencia (My) y rigidez elástica (Ke).

Estos parámetros se calculan igual que para vigas RBS.

Momento pico (Mu).*

$$M_u^{*} = a^{*} M_y$$

$$a^{*} = 9.5\left(\frac{h}{t_w}\right)^{-0.4}\left(\frac{L_b}{r_y}\right)^{-0.16}\left(1-\frac{P_g}{P_{ye}}\right)^{0.2}$$

$$1.0 \le a^{*} < 1.3$$

$a^{*} = coeficiente\ que\ depende\ de\ la\ esbeltez\ y\ fuerza\ axial$

$P_g = fuerza\ de\ compresión\ axial\ debido\ a\ cargas\ gravitacionales$

$P_{ye} = R_y F_y A = resistencia\ a\ la\ fluencia\ esperada\ de\ la\ columna$

$A = área\ de\ la\ sección\ transversal$

Rotación pre-pico (θp).*

$$\theta_p^{*} = 15\left(\frac{h}{t_w}\right)^{-1.6}\left(\frac{L_b}{r_y}\right)^{-0.3}\left(1-\frac{P_g}{P_{ye}}\right)^{2.3} \le 0.10$$

Rotación plástica post-pico (θpc).*

$$\theta_{pc}^{*} = 14\left(\frac{h}{t_w}\right)^{-0.8}\left(\frac{L_b}{r_y}\right)^{-0.5}\left(1-\frac{P_g}{P_{ye}}\right)^{3.2} \leq 0.10$$

Momento residual (Mr).*

$$M_r^{*} = \left(0.4 - 0.4\frac{P_g}{P_{ye}}\right)M_y^{*}$$

$$P_g/P_{ye} \leq 0.20, \quad M_y^{*} = 1.15ZR_yF_y\left(1-P_g/P_{ye}\right)$$

$$P_g/P_{ye} > 0.20, \quad M_y^{*} = 1.15ZR_yF_y\left[\frac{9}{8}\left(1-P_g/P_{ye}\right)\right]$$

$M_y{}^{*} = momento\ de\ fluencia$

Rotación última (θult).*

$$\theta_{ult}^{*} = 0.08\left(1-0.6\frac{P_g}{P_{ye}}\right)$$

Códigos desarrollados en MATLAB y generación de ficheros Tcl para OpenSees

Códigos originales.

Runner.

Este es el código principal del modelo ya que utiliza todas las subrutinas para ejecutar el análisis. Runner es una función que emplea distintos parámetros y define todas las propiedades de los elementos estructurales para diferentes configuraciones de edificios. Las características que se definen son los tamaños de las vigas, columnas, placas y otros detalles estructurales que se requieren para el análisis y diseño del edificio de acero en cuestión.

La función Runner tiene ocho parámetros de entrada, entre ellos se encuentran:

- Número de pisos (NumS).
- Longitud de los vanos (bay) en pulgadas (in).
- Altura del primer piso (h1) en pulgadas.
- Altura de entrepiso (hi) en pulgadas.
- Esfuerzo de fluencia del acero (Fy) en kilolibras por pulgada cuadrada (ksi).
- Factro de esfuerzo de fluencia probable (Ry).

El procedimiento que sigue la función Runner es el siguiente:

1. Se determina Ry en base al valor de Fy.
2. Se emplea una declaración de cambio para establecer los tamaños de los componentes estructurales para el NumS especificado. Para esto, se definieron cinco posibles casos de NumS. Es decir, el programa sirve para edificios de dos, cuatro, ocho, doce o veinte pisos. Para cada caso, se establecen las propiedades de los perfiles para vigas, columnas y placas. Después, los perfiles se guardan en matrices, donde cada elemento de la matriz representa el tamaño de un componente en un piso específico. Estos peraltes se calculan en base a los valores de NumS, bay, h1 y hi.
3. Se determinan las características de las placas dobles en función de los parámetros de entrada. Estas propiedades se almacenan en matrices distintas para las placas externas e internas.
4. Todas las propiedades de los distintos elementos estructurales son utilizadas para llevar a cabo el análisis estructural no lineal con la configuración especificada.

Nodes.

Esta función se encarga de crear el archivo de texto Nodes en formato Tcl, dicho fichero posee las coordenadas de los nodos para el modelo del edifico. Posteriormente, se utiliza este

archivo como entrada para OpenSees. La función considera distintos parámetros de entrada que describen la geometría de la estructura, como la cantidad de pisos, la longitud de los vanos, las alturas de los pisos, los perfiles de columnas y vigas, entre otros.

El archivo de texto que se genera cuenta con la información de las posiciones de los nodos en el plano. Estos nodos son puntos de referencia que se utilizan para establecer la localización y geometría de los componentes estructurales. Este código emplea bucles para calcular las coordenadas de los nodos de cada piso del edificio. Además, se hacen uso de convenciones de nomenclatura para asignar identificadores únicos a cada nodo, lo que brinda un sencillo reconocimiento de su ubicación en el modelo estructural.

Elements.

Esta función escribe los componentes del edificio en el archivo de texto "Elements.tcl", dichos elementos comprenden a las vigas, columnas y paneles zonales. El código Elements considera algunos datos de entrada, entre ellos:

- Número de pisos.
- Área transversal de las columnas exteriores (extcol_A) en pulgadas cuadradas (in^2).
- Área transversal de las columnas interiores (incol_A) en pulgadas cuadradas (in^2).
- Inercia respecto al eje fuerte de las columnas exteriores (extcol_Ix) en pulgadas a la cuarta (in^4).
- Inercia respecto al eje fuerte de las columnas interiores (incol_Ix) en pulgadas a la cuarta (in^4).
- Área transversal de las vigas (beam_A) en pulgadas cuadradas (in^2).
- Inercia respecto al eje fuerte de las vigas (beam_Ix) en pulgadas a la cuarta (in^4).

Con las entradas anteriores se calculan las longitudes y divisiones de los componentes estructurales. Adicionalmente, se establece una matriz “pos_splices” que tiene los índices de los pisos donde existen empalmes de columna. En resumen, la función Elements recorre cada piso de la estructura y define, etiquetando de manera única, los elementos de columnas, vigas y paneles zonales. Por último, el archivo generado se usa para OpenSees.

Zero lenght.

Este código de MATLAB crea el documento “ZeroLengthElem.tcl” que posee comandos para crear elementos de longitud nula y constraints con el mismo grado de libertad para el modelo de análisis estructural. Esto se realiza para los resortes en las partes superior e inferior de columnas, extremos de vigas, empalmes de columnas y paneles zonales. Es decir, los resortes tienen longitud cero y se definen los constraints para tomar en cuenta que dos nodos distintos, pero con las mismas coordenadas, se desplacen/roten como si fueran uno solo.

Springs.

Esta subrutina toma como datos de entrada el número de pisos; el esfuerzo de fluencia del acero; las propiedades de las secciones de las columnas exteriores, columnas interiores y vigas; rotación pre-pico, rotación post-pico, esbeltez, rigidez elástica, momento de fluencia de columnas exteriores, columnas interiores y vigas; placas doble exteriores e interiores; y la reducción de capacidad exterior e interior.

Con todos los datos mencionados anteriormente, se definen las propiedades de los resortes para las rótulas plásticas de columnas, vigas y paneles zonales de cada piso. Al final se genera el fichero “Material.tcl” con las propiedades calculadas y se utiliza este archivo para OpenSees.

Masses.

El código en cuestión crea el archivo Tcl llamado Masses, este fichero sirve para determinar y asignar las masas nodales al modelo estructural. La función realiza el siguiente procedimiento:

1. Se abre el archivo de salida y se establecen una serie de líneas de comentarios y configuraciones. Asimismo, se define el valor de la aceleración debido a la gravedad y ciertas constantes relacionadas a los pesos de los pisos.
2. Con un bucle se asignan los pesos de los pisos al modelo de cálculo. En este paso, la variable "Wtotal" guarda el peso total de la estructura con la siguiente ecuación:

$$W_{total} = \Sigma W_{piso\ i} + W_{techo} + W_{piso\ 1}$$

   Estos pesos se determinan empleando una estructura de control condicional "if" que cambia el dato del peso dependiendo si se trata del primer, último o un entrepiso.
3. Se calculan las masas nodales de cada piso. La masa nodal se determina separando el peso del piso en dos componentes, una exterior y otra interior, después se divide cada componente para la aceleración.
4. Se realiza el fichero de tal forma que las masas nodales se agreguen al modelo. Además, se hace uso de un bucle anidado para generar cuatro masas nodales por piso, cada una de ellas con un identificador único. Estos identificadores dependen del número de piso y un índice que varía de uno a cuatro. Por último, cada masa nodal es asignada a la componente exterior o interior, dependiendo de su localización en el piso.
5. La función restringe el desplazamiento lateral de los nodos del edificio. Para esto, "equalDOF" prohíbe el movimiento entre dos o más nodos. Finalmente, se utiliza un bucle anidado para establecer restricciones de movimiento en cada piso y para cada par de nodos.

Reduction capacity.

La función en cuestión calcula los factores de reducción de capacidad para las columnas sometidas a fuerzas axiales y momentos flectores. Los argumentos de entrada son el número de pisos, área transversal de las columnas exteriores e interiores, y el esfuerzo de fluencia del material. Este código sirve para diferenciar que en el método "Nonlinear Static Analysis – No Column Reduction" del App Designer no se considera la carga axial en las columnas, mientras que en el "Nonlinear Static Analysis – Column Reduction" si se toma en cuenta estas fuerzas y por tanto la capacidad de las columnas disminuye.

Códigos modificados.

Gravity generator.

Esta subrutina se enlaza con el análisis gravitacional de la estructura. Las secciones principales del procedimiento que realiza la función son las siguientes:

1. Definir una nomenclatura para identificar a los elementos estructurales del edificio.
2. Establecer el tipo de análisis que se va a ejecutar, para este caso, el análisis gravitacional.
3. Especificar los parámetros y características del edificio, aquí se incluye la cantidad de pisos, número de vanos, altura del primer piso, altura de entrepiso, restricciones de los nodos, etc.
4. Aplicar las cargas gravitacionales al modelo estructural, registrar las demandas en columnas y guardar los valores de las reacciones de la base.
5. Calcular los diferentes modos y frecuencias de vibración del edificio.
6. Registrar todas las respuestas estructurales en un archivo de salida.

Gravity general generator.

Este código de MATLAB sirve para varios propósitos, a continuación, se presenta una explicación general de cada componente:

1. Apertura del archivo “GravityGeneral.tcl” donde se almacenarán los comandos que serán ejecutados en OpenSees.
2. Redacción de líneas de comentarios y asignación de identificadores a los elementos estructurales.
3. Creación del directorio “Gravity-Analysis” para guardar los resultados que se obtengan.
4. Definición de geometría, nodos, masas y restricciones del modelo estructural.
5. Establecer los elementos de vigas y columnas con las propiedades de los resortes.
6. Aplicación de cargas gravitacionales y análisis estático.
7. Cálculo de los Eigenvalores para conseguir los diferentes periodos de vibración de la edificación.

Dynamic Tcl generator.

La función en cuestión lleva a cabo las siguientes acciones:

1. Abrir el fichero “dynamic.tcl”.
2. Redactar los comentarios y establecer las variables relacionadas con el movimiento sísmico.
3. Comenzar un bucle para iterar a través de los registros en el archivo “record.txt”.
4. En el interior del bucle, se ejecutan distintas operaciones, como restablecer el modelo estructural, aplicar el amortiguamiento al sistema, establecer parámetros de la agitación sísmica, definir patrones de carga, crear grabadores de resultados y llevar a cabo análisis dinámicos.

5. Adicionalmente, existe otro bucle que se encarga de crear comandos de grabación para las vigas y columnas en función del número de pisos que tenga la estructura.
6. Por último, se establece el paso de tiempo, restricciones para mantener el equilibrio, el algoritmo de solución y parámetros de convergencia.

Lateral generator.

En esta última subrutina, primero se define el directorio y se carga el documento "eigenvalues.out". Después se crea el fichero "Lateral.tcl" donde se encuentran las instrucciones y comandos relacionados con el modelo estructural; el método (análisis no lineal estático o análisis no lineal dinámico); las cargas de gravedad; la asignación de nodos, masas y restricciones; y el análisis de los Eigenvalores. Además, la función Lateral Generator registra los resultados de las demandas estructurales.

Aplicación creada en App Designer

La aplicación es muy sencilla de usar, solo es necesario ingresar los datos relacionados a la geometría y material de la estructura para que el programa ejecute los análisis. A continuación, se presentan los diferentes componentes de la aplicación y diagramas de flujo para visualizar como operan cada tipo de análisis.

Componentes del programa.

Número de pisos.

Para escoger el número de pisos del edificio se utilizó una lista desplegable.

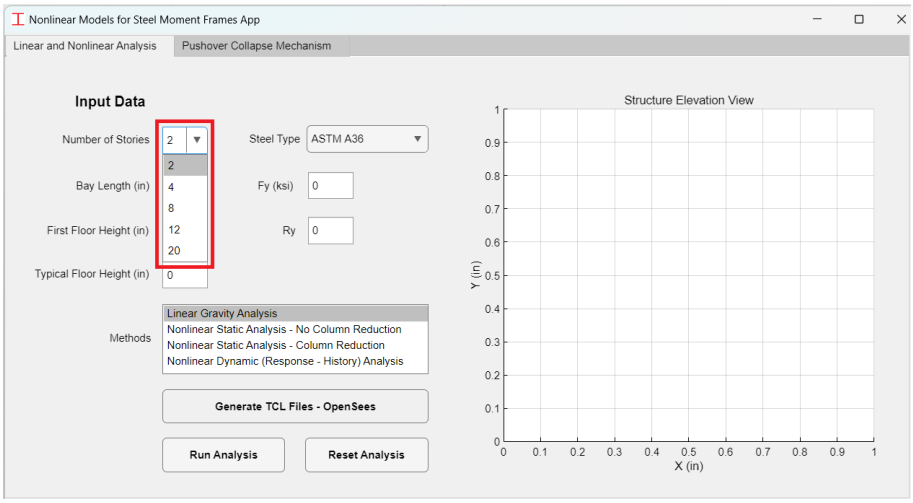

Figura 7. Lista desplegable para escoger el número de pisos.

Longitud de vanos y altura de pisos.

La longitud de los vanos, la altura del primer piso y la altura de entrepiso se ingresan en campos editables numéricos.

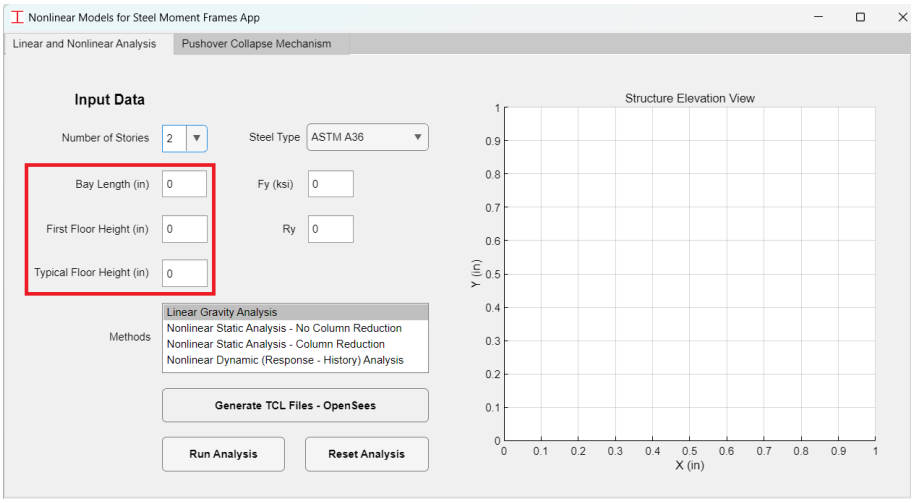

Figura 8. Campos de edición numéricos para ingresar la longitud de vanos y altura de pisos.

Tipo de acero.

Para seleccionar el tipo de acero se creó una lista desplegable.

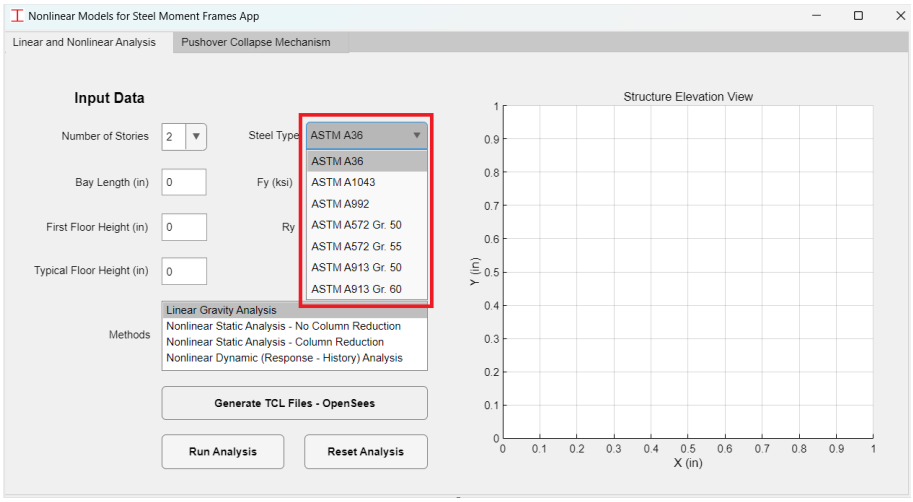

Figura 9. Lista desplegable para seleccionar el tipo de acero.

Esfuerzo de fluencia y factor de esfuerzo de fluencia probable.

En base al tipo de acero seleccionado, la aplicación muestra y asigna los valores de Fy y Ry.

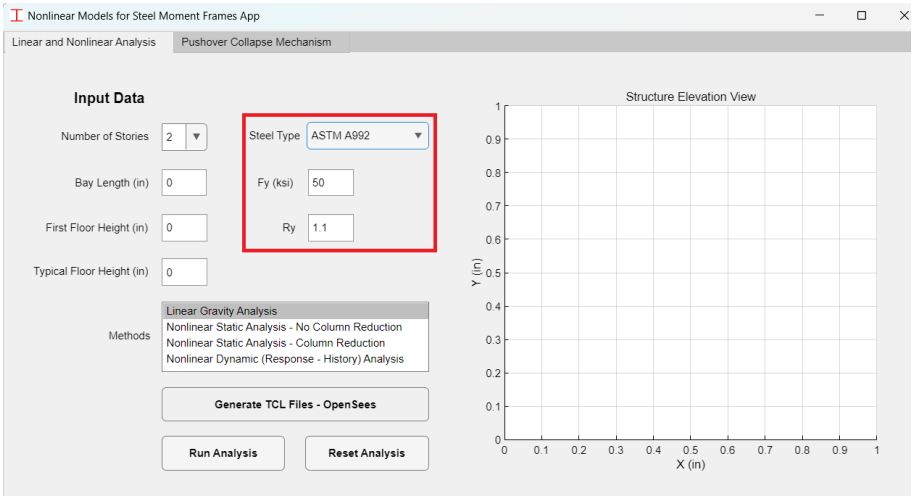

Figura 10. Valores de Fy y Ry en base al tipo de acero (ASTM A992).

Métodos.

Para escoger el tipo de análisis se empleó un cuadro de lista.

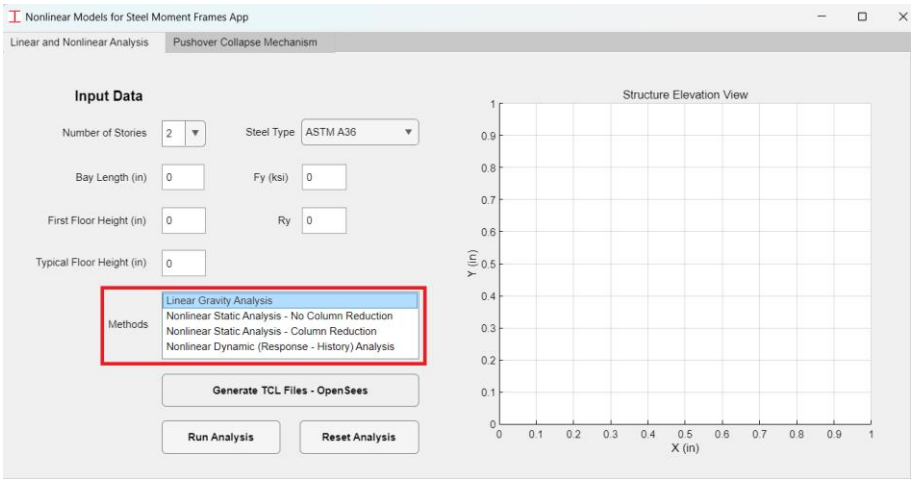

Figura 11. Cuadro de lista para escoger el tipo de análisis.

Generación de archivos Tcl y ejecución de análisis.

La creación de los ficheros Tcl y la ejecución de los análisis se realizan presionando botones.

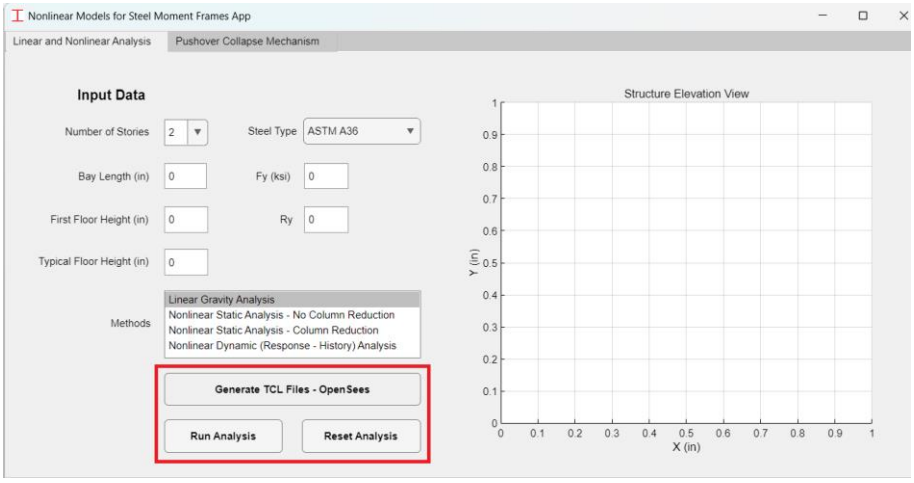

Figura 12. Botones para generar archivos Tcl, ejecutar análisis y resetear análisis.

Vista en elevación de la estructura.

Una vez que se ingresan todos los datos de entrada y se selecciona el tipo de análisis; la vista en elevación de la estructura se grafica al presionar el botón de generación de archivos Tcl.

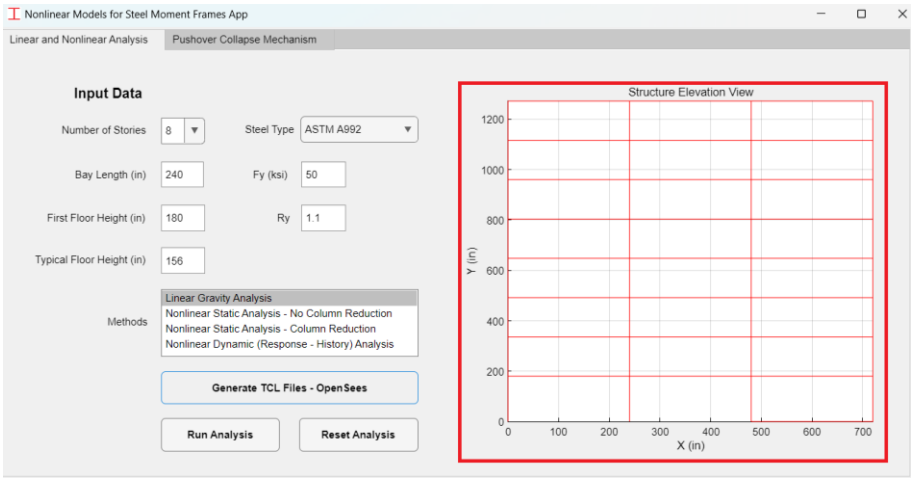

Figura 13. Vista en elevación del edificio (8 pisos).

Mecanismo de colapso del pushover.

Después de finalizar el análisis no lineal estático con reducción de columnas, es posible graficar el mecanismo de colapso en la siguiente pestaña.

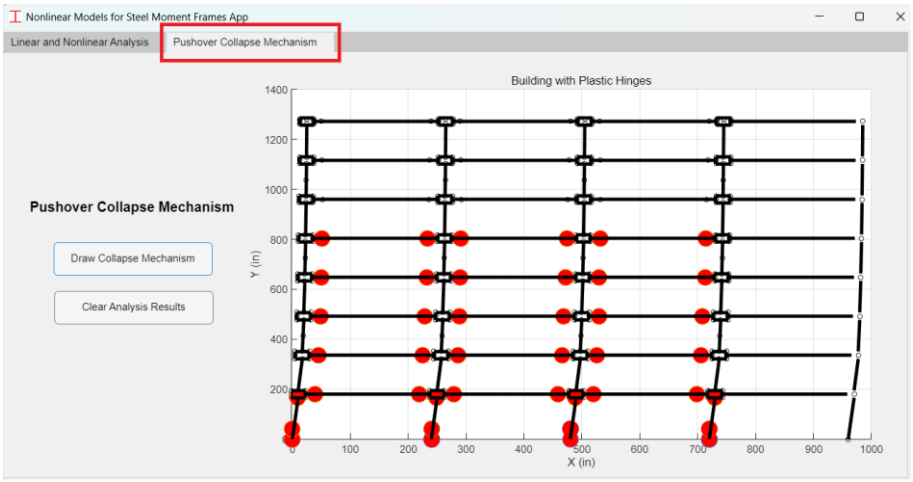

Figura 14. Mecanismo de colapso del pushover.

Diagramas de flujo de todos los tipos de análisis.

En las siguientes figuras se pueden observar las subrutinas de MATLAB, los archivos Tcl generados y los resultados que se obtienen para cada tipo de análisis.

- Análisis lineal gravitacional.

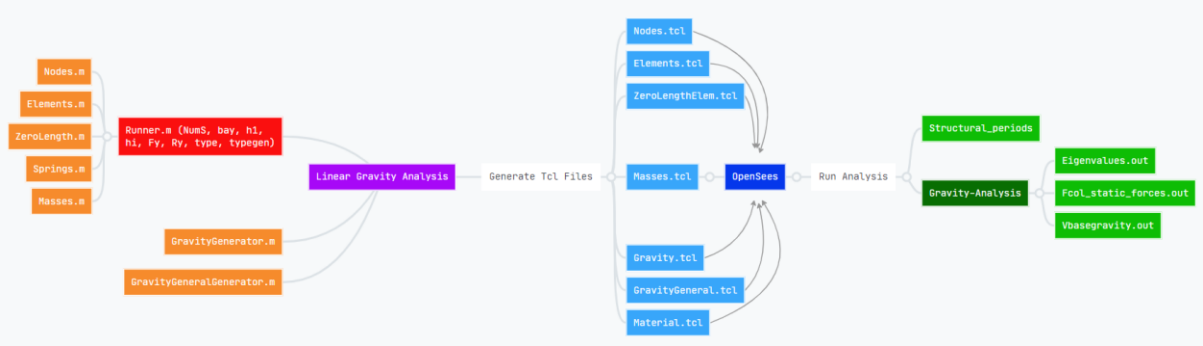

Figura 15. Diagrama de flujo del análisis lineal gravitacional.

- Análisis no lineal estático sin reducción de columnas.

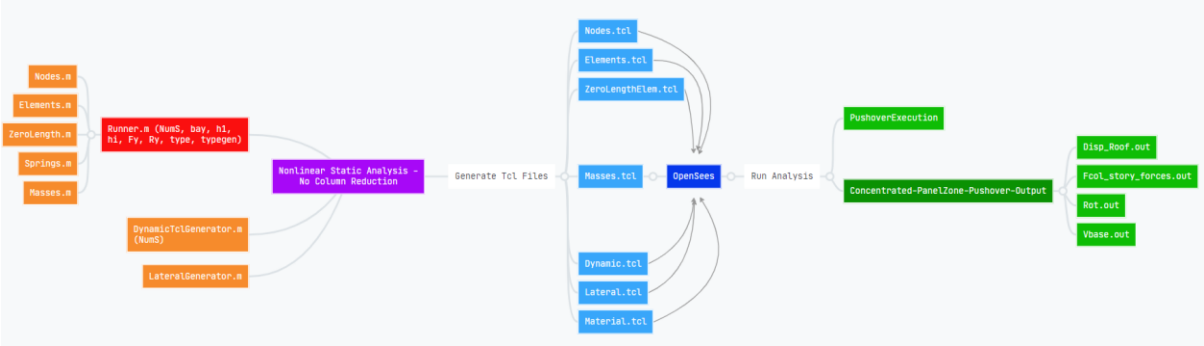

Figura 16. Diagrama de flujo del análisis no lineal estático sin reducción de columnas.

- Análisis no lineal estático con reducción de columnas.

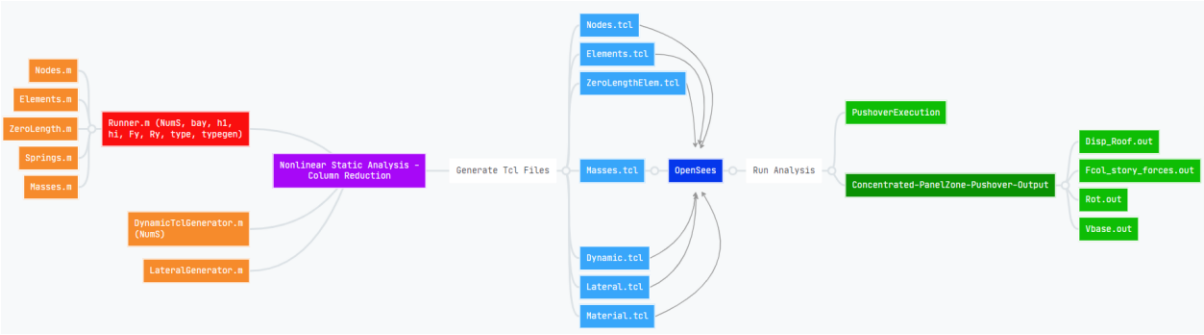

Figura 17. Diagrama de flujo del análisis no lineal estático con reducción de columnas.

- Análisis no lineal dinámico.

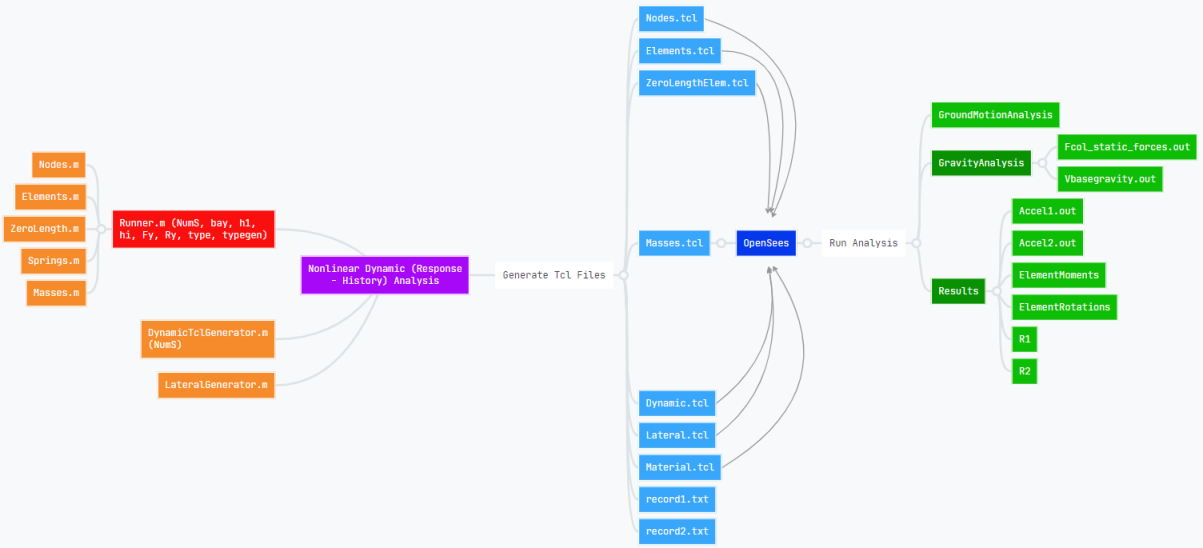

Figura 18. Diagrama de flujo del análisis no lineal dinámico.

CONCLUSIONES

Este trabajo va a ser de gran aporte para el área de ingeniería civil en el contexto ecuatoriano e internacional ya que el análisis no lineal se puede aplicar para: evaluar y diseñar soluciones de rehabilitación sísmica para edificios existentes; diseñar nuevos edificios que utilicen materiales estructurales, sistemas, u otras características que no se ajustan a los requisitos actuales de los códigos de construcción; evaluar el desempeño de los edificios para necesidades específicas de los usuarios.

En este proyecto aprendí sobre el análisis estructural no lineal para el diseño sísmico de pórticos de acero, la programación en MATLAB, el funcionamiento de OpenSees y el desarrollo de aplicaciones en App Designer. Hubo muchas dificultades en este trabajo, en especial, investigar sobre el análisis no lineal ya que es un tema que no se enseña en pregrado, por tanto, se recomienda tener sólidos conocimientos en ingeniería estructural y contar con experiencia en programación.

REFERENCIAS BIBLIOGRÁFICAS

ACI. (2014). *Building Code Requirements for Structural Concrete and Commentary*, ACI 318-14. Farmington Hills, Estados Unidos: American Concrete Institute.

AISC. (2005). *Seismic Provisions for Structural Steel Buildings*, ANSI/AISC 341-05. Chicago, Estados Unidos: American National Standards Institute/American Institute of Steel Construction.

Applied Technology Council. (2017a). *Guidelines for nonlinear structural analysis and design of buildings. Part I - general* (NIST GCR 17-917-46v1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.GCR.17-917-46v1

Applied Technology Council. (2017b). *Guidelines for nonlinear structural analysis and design of buildings. Part IIa—Steel moment frames* (NIST GCR 17-917-46v2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.GCR.17-917-46v2

ASCE. (2014). *Seismic Evaluation and Retrofit of Existing Buildings*, ASCE/SEI 41-13. Reston, Estados Unidos: American Society of Civil Engineers.

Desarrollar apps mediante App Designer—MATLAB & Simulink—MathWorks América Latina. (s/f). Recuperado el 1 de mayo de 2023, de https://la.mathworks.com/help/matlab/app-designer.html

LATBSDC. (2015). *An Alternative Procedure for Seismic Analysis and Design of Tall Buildings Located in the Los Angeles Region*. California, Estados Unidos: Los Angeles Tall Buildings Structural Design Council.

Matlab_adv.pdf. (s/f). Recuperado el 1 de mayo de 2023, de https://www-users.cse.umn.edu/~lerman/math5467/matlab_adv.pdf

NIST. (2010). *NEHRP Seismic Design Technical Brief No. 4: Nonlinear Structural Analysis for Seismic Design: A Guide for Practicing Engineers*, NIST GCR 10-917-5. Gaithersburg, Estados Unidos: Preparado por NEHRP Consultants Joint Venture, a partnership of the Applied Technology Council and the Consortium for Universities for Research in Earthquake Engineering, for the National Institute of Standards and Technology.

NIST. (2017). *Recommended Modeling Parameters and Acceptance Criteria for Nonlinear Analysis in Support of Seismic Evaluation, Retrofit, and Design*, NIST GCR 17-917-45. Gaithersburg, Estados Unidos: Applied Technology Council for the National Institute of Standards and Technology.

Nonlinear Structural Analysis For Seismic Design. (s/f). Recuperado el 1 de mayo de 2023, de https://www.nehrp.gov/pdf/nistgcr10-917-5.pdf

Open System for Earthquake Engineering Simulation—Home Page. (s/f). Recuperado el 1 de mayo de 2023, de https://opensees.berkeley.edu/

PEER. (2017). *Guidelines for Performance-Based Seismic Design of Tall Buildings*, PEER TBI. Berkeley, Estados Unidos: Pacific Earthquake Engineering Research Center Tall Buildings Initiative, College of Engineering, University of California.

PEER/ATC. (2010). *Modeling and Acceptance Criteria for Seismic Design and Analysis of Tall Building*, PEER/ATC 72-1. Redwood City, Estados Unidos: Pacific Earthquake Engineering Research Center (PEER) and Applied Technology Council (ATC).

Suzuki, Y. & Lignos, D.G. (2015). *Large scale collapse experiments of wide flange steel beam-columns.* Shanghai, China: 8th International Conference on Behavior of Steel Structures in Seismic Areas.

Tcl Developer Site. (s/f). Recuperado el 1 de mayo de 2023, de https://www.tcl.tk/

ANEXO A: SUBRUTINAS DE MATLAB

Runner

```
function Runner(NumS, bay, h1, hi, Fy, Ry, type, ~) %(NumS, bay, h1, hi, Fy, Ry,
type, ~)


%Structural configuration:

% NumS;
% bay;
% h1;
% hi;
% Ry;
% type;
% Fy;

%PROPERTIES OF THE PLASTIC HINGES:
%type=2 GRAVITY ANALYSIS!!
%type=1 REDUCTION CAPACITY OF THE COLUMN !!


%Member sizes:

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

% if Fy == 50
%     Ry = 1.1;
% else
%     Ry = 1.3;
% end

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 = 50; % 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

%Plastic hinge configuration and location

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=1.1*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,sh_ext,sh_int);
Elements(nf,NumS,extcol_A,incol_A,extcol_Ix,incol_Ix,beam_A, beam_Ix);
ZeroLength(NumS);
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,incol_bf,i
ncol_tf,beam_d,ext_col_theta_p,ext_col_theta_pc,ext_col_lambda,Ke_ext_col,My_ext_c
ol,ass_ext_col,int_col_theta_p,int_col_theta_pc,int_col_lambda,Ke_int_col,My_int_c
ol,ass_int_col,My_RBS,beam_theta_p,beam_theta_pc,beam_lambda,Ke_beam,ass_beam,ext_
dblplate,int_dblplate,reduction_capacity_ext,reduction_capacity_int);
```

```
masses(NumS);
end
```

Nodes

```
function Nodes(NumS,bay,h1,hi,beam_d,extcol_d,incol_d,sh_ext,sh_int)

% 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;

%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', '#nodeID convention:  "xy" where x = Pier # and y =
Floor #');
fprintf(fileID, '%s\r\n', '');

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, '%s\r\n', '');

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);
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '51', x+4*bay, y);

%Nodes for the springs at first floor:

%nodeID convention:  "xya" where x = Pier #, y = Floor #, a = 7
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 = 7');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '118', x, y);
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '218', x+bay, y);
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '318', x+2*bay, y);
fprintf(fileID, '%5s %5s %10.3f %10.3f\r\n', 'node ', '418', x+3*bay, y);

%COORDINATES OF THE NODES FOR THE SPRINGS AT THE COLUMNS:
```

```
%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)

 for i=2:NumS+1;

     fprintf(fileID, '%s\r\n', '');
     fprintf(fileID, '%s %u\r\n', '#NODES FOR THE SPRINGS OF THE COLUMNS AT FLOOR:
', i);
     fprintf(fileID, '%s\r\n', '#nodeID convention:  "xya" where x = Pier #, y =
Floor #, "a" convention: 5,6 = below; 7,8 = above');
     fprintf(fileID, '%s\r\n', '');

     if i==2
            y=h1;
        else
            y=y+hi;
     end
     x=0;
     for j=1:4;

        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=k+1;
        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);
        k=k+1;
        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);
        k=k+1;
        if i==NumS+1 && k==8

            fprintf(fileID, '%s\r\n', '');
        else
            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);
        end

        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=1:4;

         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 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=1:3;
             y2=h1+(NumS-1)*hi;
             if j==3;

                 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=1:3;

         if j==3;

            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


 %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)');

 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=1:4;
        if j==1 || j==4;
           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', '');
        x=x+bay;

     end
     b=b+hi;

 end

 fclose(fileID);


end
```

Elements

```
function Elements(nf,NumS,extcol_A,incol_A,extcol_Ix,incol_Ix,beam_A, beam_Ix)

% 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;

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

%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

num_splices=z;
k=1;
p=1;
q=1;
for i=1:NumS;

     if i==pos_splices(i);
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN 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=1:4;
            for l=91:92;
                if j==1 || j==4;
                    if l==91;

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),'8']); %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)*(n+1)/n, '$transfTag');
                        a=str2num(ColumnTag);
                        col(p)=a;
                    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),'5']);
                        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)*(n+1)/n, '$transfTag');
                        b=str2num(ColumnTag);
                        col(p)=b;
                    end
                else

                    if l==91;

                        ColumnTag=(['1',num2str(j),num2str(i),num2str(l)]);
                        node1 = ([num2str(j), num2str(i),'8']); %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)*(n+1)/n, '$transfTag');
                        c=str2num(ColumnTag);
                        col(p)=c;
                    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),'5']);
                        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)*(n+1)/n, '$transfTag');
                        d=str2num(ColumnTag);
                        col(p)=d;
                    end

                end
                p=p+1;
            end

        end


    else

        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#COLUMN 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=1:4;

            if j==1 || j==4;

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

```
                node1 = ([num2str(j), num2str(i),'8']); %num2str = converts a
numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'5']);
                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)*(n+1)/n, '$transfTag');
                e=str2num(ColumnTag);
                col2(q)=e;
            else

                ColumnTag=(['1',num2str(j),num2str(i)]);
                node1 = ([num2str(j), num2str(i),'8']); %num2str = converts a
numeric array into a character array
                node2 = ([num2str(j), num2str(i+1),'5']);
                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)*(n+1)/n, '$transfTag');
                f=str2num(ColumnTag);
                col2(q)=f;
            end
            q=q+1;
         end
        k=k+1;

     end

end

% a1=size(col);
% a2=size(col2);
% a3=a1(1,2);
% a4=a2(1,2);
% a5=a3+a4;
%
% for i=1:a5
%  if (i<=a3)
%     col3(i)=col(i);
%  else
%      col3(i)=col2(i-a3);
%  end
% end

% col3;
% region_col=num2str(col3);

if NumS==2
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%14s %12s\r\n', 'region 4 -ele     111 121 131 141 112 122 132
142');
fprintf(fileID, '%s\r\n', '');
end

if NumS==4
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%14s %12s\r\n', 'region 4 -ele  111      121    131    141    112
      122    132    142    11391  11392  12391  12392  13391  13392  14391  14392  114
      124    134    144');
fprintf(fileID, '%s\r\n', '');
```

```
end

if NumS==8
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%14s %12s\r\n', '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');
fprintf(fileID, '%s\r\n', '');
end

if NumS==12
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%14s %12s\r\n', '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    11991  11992  12991  12992  13991  13992  14991  14992  1110
      1210   1310   1410   111191 111192 121191 121192 131191 131192 141191 141192 1112
      1212   1312   1412');
fprintf(fileID, '%s\r\n', '');
end

if NumS==20
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%14s %12s\r\n', '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    11991  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');
fprintf(fileID, '%s\r\n', '');
end

%BEAM ELEMENTS:

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=1:3;

            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=1:4;


            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),'6']);
            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),'6']); %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
```

Zero length

```
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', '');

fprintf(fileID, '%s\r\n', '# ZERO LENGTH ELEMENTS FOR COLUMN BASE CONNECTIONS');
fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n', 'element
zeroLength', '1', '11', '117', '-mat ', '666 666 331', '-dir ','1 2 6');
fprintf(fileID, '%s\r\n', '');
```

```
fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n', 'element
zeroLength', '2', '21', '217', '-mat ', '666 666 331', '-dir ','1 2 6');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n', 'element
zeroLength', '3', '31', '317', '-mat ', '666 666 331', '-dir ','1 2 6');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n', 'element
zeroLength', '4', '41', '417', '-mat ', '666 666 331', '-dir ','1 2 6');
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:NumS;

     if i==pos_splices(i);
        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', '');
         for j=1:4;

              ColumnTag1=(['3',num2str(j),num2str(i),'1']);
              node1 = ([num2str(j), num2str(i),'7']); %num2str = converts a
numeric array into a character array
              node2 = ([num2str(j), num2str(i),'8']);
              fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', ColumnTag1, node1, node2, '-mat ', ColumnTag1, '-dir ',
'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', '');

              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 ', ColumnTag2, '-dir ',
'6');
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', node3,
node4, '1', '2');
            fprintf(fileID, '%s\r\n', '');

            ColumnTag3=(['3',num2str(j),num2str(i),'2']);
            node5 = ([num2str(j), num2str(i+1),'5']); %num2str = converts a
numeric array into a character array
            node6 = ([num2str(j), num2str(i+1),'6']);
            fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n',
'element zeroLength', ColumnTag3, node5, node6, '-mat ', ColumnTag3, '-dir ',
'6');
            fprintf(fileID, '%s\r\n', '');
            fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', node5,
node6, '1', '2');
            fprintf(fileID, '%s\r\n', '');



         end


     else

        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');
        fprintf(fileID, '%s\r\n', '');

         for j=1:4;

           ColumnTag1=(['3',num2str(j),num2str(i),'1']);
           node1 = ([num2str(j), num2str(i),'7']); %num2str = converts a numeric
array into a character array
           node2 = ([num2str(j), num2str(i),'8']);
           fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n', 'element
zeroLength', ColumnTag1, node1, node2, '-mat ', ColumnTag1, '-dir ', '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', '');

           ColumnTag2=(['3',num2str(j),num2str(i),'2']);
           node3 = ([num2str(j), num2str(i+1),'5']); %num2str = converts a numeric
array into a character array
           node4 = ([num2str(j), num2str(i+1),'6']);
           fprintf(fileID, '%18s %10s %6s %6s %6s %6s %10s %6s %12s\r\n', 'element
zeroLength', ColumnTag2, node3, node4, '-mat ', ColumnTag2, '-dir ', '6');
           fprintf(fileID, '%s\r\n', '');
           fprintf(fileID, '%8s %6s %6s %6s %12s\r\n', 'equalDOF', node3, node4,
'1', '2');
```

```
            fprintf(fileID, '%s\r\n', '');


        end
        k=k+1;
    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', '');
            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
fclose(fileID);
end
```

Springs

```
function
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf,incol_d,incol_tw,incol_bf,i
ncol_tf,beam_d,ext_col_theta_p,ext_col_theta_pc,ext_col_lambda,Ke_ext_col,My_ext_c
ol,ass_ext_col,int_col_theta_p,int_col_theta_pc,int_col_lambda,Ke_int_col,My_int_c
ol,ass_int_col,My_RBS,beam_theta_p,beam_theta_pc,beam_lambda,Ke_beam,ass_beam,ext_
dblplate,int_dblplate,reduction_capacity_ext,reduction_capacity_int)


fileID = fopen(['Material','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# PROPERTIES OF SPRINGS FOR THE PLASTIC HINGES');
fprintf(fileID, '%s\r\n', '');
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;


for i=1:NumS;
     %k=1;
     if i==pos_splices(i);
        fprintf(fileID, '%s\r\n', '');
        fprintf(fileID, '%s %u\r\n', '#MATERIAL PROPERTIES FOR COLUMN SPRINGS AT
STORY: ', i);
        fprintf(fileID, '%s\r\n', '');

         for j=1:4;

           if j==1 || j==4

              ColumnTag1=(['3',num2str(j),num2str(i),'1']);
              ke = num2str(Ke_ext_col(i));
              as = num2str(reduction_capacity_ext(i)*ass_ext_col(i));
              My= num2str(reduction_capacity_ext(i)*My_ext_col(i));
              My2=num2str(-reduction_capacity_ext(i)*My_ext_col(i));
              Lamda= num2str(ext_col_lambda(i));
              Cc=num2str(1);
              theta_p=num2str(ext_col_theta_p(i));
              theta_pc=num2str(ext_col_theta_pc(i));
              res=num2str(0.4);
              theta_u=num2str(0.2);
              D=num2str(1);

              fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag1, 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', '');

              ColumnTag2=(['3',num2str(j),num2str(i),'3']);
              ke = num2str(Ke_ext_col(i+1));
              as = num2str(reduction_capacity_ext(i)*ass_ext_col(i+1));
              My= num2str(reduction_capacity_ext(i)*My_ext_col(i+1));
              My2=num2str(-reduction_capacity_ext(i)*My_ext_col(i+1));
              Lamda= num2str(ext_col_lambda(i+1));
              Cc=num2str(1);
              theta_p=num2str(ext_col_theta_p(i+1));
              theta_pc=num2str(ext_col_theta_pc(i+1));
              res=num2str(0.4);
              theta_u=num2str(0.2);
              D=num2str(1);
```

```
            fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag2, 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', '');

            ColumnTag3=(['3',num2str(j),num2str(i),'2']);
            ke = num2str(Ke_ext_col(i+1));
            as = num2str(reduction_capacity_ext(i)*ass_ext_col(i+1));
            My= num2str(reduction_capacity_ext(i)*My_ext_col(i+1));
            My2=num2str(-reduction_capacity_ext(i)*My_ext_col(i+1));
            Lamda= num2str(ext_col_lambda(i+1));
            Cc=num2str(1);
            theta_p=num2str(ext_col_theta_p(i+1));
            theta_pc=num2str(ext_col_theta_pc(i+1));
            res=num2str(0.4);
            theta_u=num2str(0.2);
            D=num2str(1);

            fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag3, 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', '');

        else

            ColumnTag1=(['3',num2str(j),num2str(i),'1']);
            ke = num2str(Ke_int_col(i));
            as = num2str(reduction_capacity_int(i)*ass_int_col(i));
            My= num2str(reduction_capacity_int(i)*My_int_col(i));
            My2=num2str(-reduction_capacity_int(i)*My_int_col(i));
            Lamda= num2str(int_col_lambda(i));
            Cc=num2str(1);
            theta_p=num2str(int_col_theta_p(i));
            theta_pc=num2str(int_col_theta_pc(i));
            res=num2str(0.4);
            theta_u=num2str(0.2);
            D=num2str(1);

            fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag1, 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', '');

            ColumnTag2=(['3',num2str(j),num2str(i),'3']);
            ke = num2str(Ke_int_col(i+1));
            as = num2str(reduction_capacity_int(i)*ass_int_col(i+1));
            My= num2str(reduction_capacity_int(i)*My_int_col(i+1));
            My2=num2str(-reduction_capacity_int(i)*My_int_col(i+1));
            Lamda= num2str(int_col_lambda(i+1));
            Cc=num2str(1);
            theta_p=num2str(int_col_theta_p(i+1));
            theta_pc=num2str(int_col_theta_pc(i+1));
            res=num2str(0.4);
            theta_u=num2str(0.2);
```

```
            D=num2str(1);


            fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag2, 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', '');

            ColumnTag3=(['3',num2str(j),num2str(i),'2']);
            ke = num2str(Ke_int_col(1+i));
            as = num2str(reduction_capacity_int(i)*ass_int_col(1+i));
            My= num2str(reduction_capacity_int(i)*My_int_col(1+i));
            My2=num2str(-reduction_capacity_int(i)*My_int_col(1+i));
            Lamda= num2str(int_col_lambda(1+i));
            Cc=num2str(1);
            theta_p=num2str(int_col_theta_p(1+i));
            theta_pc=num2str(int_col_theta_pc(1+i));
            res=num2str(0.4);
            theta_u=num2str(0.2);
            D=num2str(1);

            fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag3, 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



        end
        %k=k+1;

    else

       fprintf(fileID, '%s\r\n', '');
       fprintf(fileID, '%s %u\r\n', '#MATERIAL PROPERTIES FOR COLUMN SPRINGS AT
STORY: ', i);
       fprintf(fileID, '%s\r\n', '');

        for j=1:4;
           if j==1 || j==4
            ColumnTag1=(['3',num2str(j),num2str(i),'1']);
            ke = num2str(Ke_ext_col(i));
            as = num2str(reduction_capacity_ext(i)*ass_ext_col(i));
            My= num2str(reduction_capacity_ext(i)*My_ext_col(i));
            My2=num2str(-reduction_capacity_ext(i)*My_ext_col(i));
            Lamda= num2str(ext_col_lambda(i));
            Cc=num2str(1);
            theta_p=num2str(ext_col_theta_p(i));
            theta_pc=num2str(ext_col_theta_pc(i));
            res=num2str(0.4);
            theta_u=num2str(0.2);
            D=num2str(1);
```

```
                fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag1, 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', '');

                ColumnTag3=['3',num2str(j),num2str(i),'2'];
                ke = num2str(Ke_ext_col(i));
                as = num2str(reduction_capacity_ext(i)*ass_ext_col(i));
                My= num2str(reduction_capacity_ext(i)*My_ext_col(i));
                My2=num2str(-reduction_capacity_ext(i)*My_ext_col(i));
                Lamda= num2str(ext_col_lambda(i));
                Cc=num2str(1);
                theta_p=num2str(ext_col_theta_p(i));
                theta_pc=num2str(ext_col_theta_pc(i));
                res=num2str(0.4);
                theta_u=num2str(0.2);
                D=num2str(1);

                fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag3, 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', '');

            else

                ColumnTag1=['3',num2str(j),num2str(i),'1'];
                ke = num2str(Ke_int_col(i));
                as = num2str(reduction_capacity_int(i)*ass_int_col(i));
                My= num2str(reduction_capacity_int(i)*My_int_col(i));
                My2=num2str(-reduction_capacity_int(i)*My_int_col(i));
                Lamda= num2str(int_col_lambda(i));
                Cc=num2str(1);
                theta_p=num2str(int_col_theta_p(i));
                theta_pc=num2str(int_col_theta_pc(i));
                res=num2str(0.4);
                theta_u=num2str(0.2);
                D=num2str(1);

                fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag1, 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', '');

                ColumnTag3=['3',num2str(j),num2str(i),'2'];
                ke = num2str(Ke_int_col(i));
                as = num2str(reduction_capacity_int(i)*ass_int_col(i));
                My= num2str(reduction_capacity_int(i)*My_int_col(i));
                My2=num2str(-reduction_capacity_int(i)*My_int_col(i));
                Lamda= num2str(int_col_lambda(i));
                Cc=num2str(1);
                theta_p=num2str(int_col_theta_p(i));
                theta_pc=num2str(int_col_theta_pc(i));
                res=num2str(0.4);
                theta_u=num2str(0.2);
                D=num2str(1);
```

```
              fprintf(fileID, '%24s %6s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s
%2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s %2s', 'uniaxialMaterial  Bilin ',
ColumnTag3, 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

        end

    end

end



%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', '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', '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', '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);

end
```

Masses

```
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

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



end
```

Reduction capacity

```
function [factor_ext,factor_int]=reduction_capacity(NumS,extcol_A,incol_A,Fy)

aux_dir=pwd;
cambio=[aux_dir '\Concentrated-PanelZone-Pushover-Output'];
cd(cambio);

Forces=load('Fcol_story_forces.out');

cd ..

cambio=[aux_dir '\Gravity-Analysis'];
cd(cambio);

Static=load('Fcol_static_forces.out');
m=size(Forces);

col=m(1,2);
row=m(1,1);
a=2;

for j=1:col
    for i=1:row

        Vector_Push(i,j)=Forces(i,a);
```

```
    end
    a=a+6;

    if a>col
        break
    end

end

c=2;
for j=1:col

    Vector_Static(1,j)=Static(1,c);
    c=c+6;

    if c>col
        break
    end

end

MaxPush=max(abs(Vector_Push));
Pure_Push=abs(MaxPush)-abs(Vector_Static);

Axial=Vector_Static+0.5*abs(Pure_Push);
NumCol=4*NumS;

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;

data=(NumS+num_splices)*4;

for i=1:data
   pos(i)=i;

end


Ext_capacity=Fy*1.1*extcol_A;
Int_capacity=Fy*1.1*incol_A;
Ext_capacity2=Ext_capacity;
Int_capacity2=Int_capacity;
```

```
s1=size(Axial);
s2=s1(1,2);


a=1;

for j=1:NumS
 if j==pos_splices(j)
   Col_capacity(a)=Ext_capacity(j);
   a=a+1;
   Col_capacity(a)=Ext_capacity(j+1);
   a=a+1;
   Col_capacity(a)=Int_capacity(j);
   a=a+1;
   Col_capacity(a)=Int_capacity(j+1);
   a=a+1;
   Col_capacity(a)=Int_capacity2(j);
   a=a+1;
   Col_capacity(a)=Int_capacity2(j+1);
   a=a+1;
   Col_capacity(a)=Ext_capacity2(j);
   a=a+1;
   Col_capacity(a)=Ext_capacity2(j+1);
   a=a+1;


 else
   Col_capacity(a)=Ext_capacity(j);
   a=a+1;
   Col_capacity(a)=Int_capacity(j);
   a=a+1;
   Col_capacity(a)=Int_capacity2(j);
   a=a+1;
   Col_capacity(a)=Ext_capacity2(j);
   a=a+1;
 end
end

 for i=1:s2
    ratio=Axial(i)/Col_capacity(i);
    if ratio>0.2
        factor2(i)=(9/8)*(1-ratio);
    else
        factor2(i)=(1-ratio/2);
    end
 end


a1=8;   %First two stories

for i=1:a1
    B(i)=factor2(i);
end

for i=1:num_splices

    B(8+8*(i-1)+1)=factor2(8+12*(i-1)+1);
    B(8+8*(i-1)+2)=factor2(8+12*(i-1)+3);
```

```
    B(8+8*(i-1)+3)=factor2(8+12*(i-1)+5);
    B(8+8*(i-1)+4)=factor2(8+12*(i-1)+7);


    B(8+8*(i-1)+5)=factor2(8+12*(i-1)+9);
    B(8+8*(i-1)+6)=factor2(8+12*(i-1)+10);
    B(8+8*(i-1)+7)=factor2(8+12*(i-1)+11);
    B(8+8*(i-1)+8)=factor2(8+12*(i-1)+12);

end

 b=4;
 a=1;
 c=3;

 while b<=s2-4*num_splices
    factor_ext(a)=B(b);
    factor_int(a)=B(c);
    b=b+4;
    a=a+1;
    c=c+4;
 end

 cd ..
 end
```

Gravity generator

```
fileID = fopen(['Gravity','.tcl'],'w');


fprintf(fileID, '%s\r\n', '# Element ID conventions:');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#1xy    = frame columns with RBS springs at both
ends');
fprintf(fileID, '%s\r\n', '#2xy    = frame beams with RBS springs at both ends');
fprintf(fileID, '%s\r\n', '#6xy    = trusses linking frame and P-delta column');
fprintf(fileID, '%s\r\n', '#7xy    = P-delta columns');
fprintf(fileID, '%s\r\n', '#2xya   = frame beams between panel zone and RBS
spring');
fprintf(fileID, '%s\r\n', '#3xya   = frame column rotational springs');
fprintf(fileID, '%s\r\n', '#4xya   = frame beam rotational springs');
fprintf(fileID, '%s\r\n', '#5xya   = P-delta column rotational springs');
fprintf(fileID, '%s\r\n', '#4xy00  = panel zone rotational springs');
fprintf(fileID, '%s\r\n', '#500xya = panel zone elements');
fprintf(fileID, '%s\r\n', '#where:');
fprintf(fileID, '%s\r\n', '#x = Pier or Bay #');
fprintf(fileID, '%s\r\n', '#y = Floor or Story #');
fprintf(fileID, '%s\r\n', '#a = an integer describing the location relative to
beam-column joint (see description where elements and nodes are defined)');

fprintf(fileID, '%s\r\n',
'##############################################################################
##################','#          Set Up & Source
Definition','##############################################################################
##############################');
```

```
fprintf(fileID, '%s\r\n','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');
fprintf(fileID, '%s\r\n',
'################################################################################
##################','#          Define Analysis
Type','################################################################################
########################');
fprintf(fileID, '%s\r\n', 'set dataDir Gravity-Analysis;','file mkdir $dataDir;');
fprintf(fileID, '%s\r\n',
'################################################################################
##################','#          Define Building, Geometry, Nodes, Masses and
Constraints','################################################################################
################################');
fprintf(fileID, '%s\r\n', 'set n10;',' ');

fprintf(fileID, '%s%d\r\n', 'set NStories ', NumS);
fprintf(fileID, '%s\r\n', 'set NBays 3');
fprintf(fileID, '%s%d\r\n', 'set WBay ',bay);
fprintf(fileID, '%s%d\r\n', 'set HStory1 ',h1);
fprintf(fileID, '%s%d\r\n', 'set HStoryTyp ',hi);
fprintf(fileID, '%s%d\r\n', 'set HBuilding ',(h1+(NumS-1)*hi));
fprintf(fileID, '%s\r\n', 'source Nodes.tcl','source Masses.tcl');
fprintf(fileID, '%s\r\n', ' ','# assign boundary condidtions ','# command:  fix
nodeID dxFixity dyFixity rzFixity','# fixity values: 1 = constrained; 0 =
unconstrained','# fix the base of the building; pin P-delta column at base');
fprintf(fileID, '%s\r\n', '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;');
fprintf(fileID, '%s\r\n', '# P-delta column is pinned');
fprintf(fileID, '%s\r\n', 'set transfTag 1;','geomTransf PDelta $transfTag; ', '#
PDelta transformation',' ');

fprintf(fileID, '%s\r\n', 'source Elements.tcl','source Material.tcl','source
ZeroLengthElem.tcl');


fprintf(fileID, '%s\r\n','',
'################################################################################
##################','#          Gravity Loads & Gravity
Analysis','################################################################################
############################');

fprintf(fileID, '%s\r\n', '','# apply gravity loads','#command: pattern
PatternType $PatternID TimeSeriesType','pattern Plain 101 Constant {');

fprintf(fileID, '%s\r\n', '# point loads on leaning column nodes','# command: load
node Fx Fy Mz');

fprintf(fileID, '%s\r\n', 'set P_PD2 [expr -568.00];      # Floor 2','set P_PD3
[expr -565.20];    # Floor 3');

fprintf(fileID, '%s%d', 'set P_PD',NumS+1,' [expr -511.20];     # Floor
',NumS+1');

fprintf(fileID, '%s\r\n','  ',' ');
```

```
fprintf(fileID, '%s%d', 'load 5',2,' 0.0 $P_PD',2,' 0.0;# Floor ',2);
fprintf(fileID, '\n');

k=3;

for i=3:NumS
    fprintf(fileID, '%s%d', 'load 5',k,' 0.0 $P_PD',3,' 0.0;# Floor ',k);
    fprintf(fileID, '\n');


k=k+1;

end

fprintf(fileID, '%s%d', 'load 5',NumS+1,' 0.0 $P_PD',NumS+1,' 0.0;           #
Floor ',NumS+1);
fprintf(fileID, '\n');


fprintf(fileID, '%s\r\n', ' ','# point loads on frame column nodes',' ');

k=2;

for i=2:NumS
    fprintf(fileID, '%s%d','set P_F',k,'1 [expr -70.21];  # load on each frame
node in Floor ',k,' (exterior)');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','set P_F',k,'2 [expr -46.14];  # load on each frame
node in Floor ',k,' (interior)');
    fprintf(fileID, '\n');
    k=k+1;
end

fprintf(fileID, '%s%d','set P_F',NumS+1,'1 [expr -61.77]; # load on each frame
node in Floor ',NumS+1,' (exterior)');
fprintf(fileID, '\n');
fprintf(fileID, '%s%d','set P_F',NumS+1,'2 [expr -41.18]; # load on each frame
node in Floor ',NumS+1,' (interior)');
fprintf(fileID, '\n');

k=2;

fprintf(fileID, '%s\r\n','  ',' ');

for i=2:NumS+1
    fprintf(fileID, '%s%d','# Floor ',k,' loads');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 1',k,'7 0.0 $P_F',k,'1 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 2',k,'7 0.0 $P_F',k,'2 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 3',k,'7 0.0 $P_F',k,'2 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 4',k,'7 0.0 $P_F',k,'1 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s\r\n','  ',' ');
    k=k+1;
end
```

```
fprintf(fileID, '%s\r\n', '','','recorder Element -file
$dataDir/Fcol_static_forces.out -region 4 force;','recorder Node -file
$dataDir/Vbasegravity.out -node 11 21 31 41 51 -dof 1 2 3 reaction;');

fprintf(fileID, '%s\r\n', '','','     }');
fprintf(fileID, '%s\r\n', '','','# Gravity-analysis: load-controlled static
analysis');
fprintf(fileID, '%s\r\n','set Tol 1.0e-6;                       # convergence
tolerance for test');
fprintf(fileID, '%s\r\n','constraints Plain;             # how it handles
boundary conditions');
fprintf(fileID, '%s\r\n','numberer RCM;                         # renumber dofs
to minimize band-width (optimization)');
fprintf(fileID, '%s\r\n','system BandGeneral;            # how to store and solve
the system of equations in the analysis (large model: try UmfPack)');
fprintf(fileID, '%s\r\n','test NormDispIncr $Tol 6;      # determine if
convergence has been achieved at the end of an iteration step');
fprintf(fileID, '%s\r\n','algorithm Newton;              # use Newtons solution
algorithm: updates tangent stiffness at every iteration');
fprintf(fileID, '%s\r\n','set NstepGravity 1;            # apply gravity in 10
steps');
fprintf(fileID, '%s\r\n','set DGravity [expr 1.0/$NstepGravity]; # load
increment');
fprintf(fileID, '%s\r\n','integrator LoadControl $DGravity;     # determine the
next time step for an analysis');
fprintf(fileID, '%s\r\n','analysis Static;               # define type of
analysis: static or transient');
fprintf(fileID, '%s\r\n','analyze $NstepGravity;                # apply
gravity');
fprintf(fileID, '%s\r\n','','# maintain constant gravity loads and reset time to
zero');
fprintf(fileID, '%s\r\n','loadConst -time 0.0');
fprintf(fileID, '%s\r\n','puts "Model Built"','','','');

fprintf(fileID,
'%s\r\n','################################################################################
####','#                       Eigenvalue Analysis

','################################################################################','
','');

fprintf(fileID, '%s\r\n','set pi [expr 2.0*asin(1.0)];                #
Definition of pi');

allLetters='A':'Z';
for i=1:NumS
    aux_letters=['set nEigen',allLetters(i),' ', num2str(i),';'              '
# mode ',allLetters(i), '=', num2str(i)];
    fprintf(fileID,'%s\r\n', aux_letters);
end
fprintf(fileID, '%s\r\n', '');
aux_letters2=['set lambdaNN [eigen  -fullGenLapack  [expr
$nEigen',allLetters(i),']];','   # eigenvalue analysis for nEigen',allLetters(i),'
modes'];
fprintf(fileID, '%s\r\n',aux_letters2);
fprintf(fileID, '%s\r\n', '');
```

```
%fprintf(fileID, '%s\r\n','     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','','');
k=1;
for i=1:NumS
     aux_letters3=['set lambda',allLetters(i),' [lindex $lambdaNN [expr
$nEigen',allLetters(i), '-1]];','   # eigenvalue mode
',allLetters(i),'=',num2str(i)];
     fprintf(fileID, '%s%d',aux_letters3);
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');

k=1;
for i=1:NumS
     aux_letters4=[' [expr pow($lambda',allLetters(i),',0.5)]; # w',num2str(i),
'(',num2str(i), 'st mode circular frequency)'];
     fprintf(fileID, '%s%d','set w',k,aux_letters4);
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');


k=1;
for i=1:NumS

     fprintf(fileID, '%s%d','set T',k,' [expr 2.0*$pi/$w',k,']; # mode period of
the structure');
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');

k=1;

for i=1:NumS

     fprintf(fileID, '%s%d','puts "T',k,' = $T',k,' s";   # display the ',k,' mode
period in the command window');
     fprintf(fileID, '\n');
     k=k+1;
end

for i=2:NumS+1
    vector_numS{i-1}=['1',num2str(i),'005'];

end
```

```
vector_numS_aux=string(vector_numS);
aux_print=['recorder Node -file $dataDir/eigenvalues.out -node 118
',vector_numS_aux, ' -dof 1 "eigen 1"'];

aux_print=char(aux_print);
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', aux_print);
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '# record the eigenvectors');
fprintf(fileID, '%s\r\n', 'record');
fprintf(fileID, '%s\r\n','wipe all;');
```

Gravity general generator

```
fileID = fopen(['GravityGeneral','.tcl'],'w');


fprintf(fileID, '%s\r\n', '# Element ID conventions:');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#    1xy    = frame columns with RBS springs at both
ends');
fprintf(fileID, '%s\r\n', '#    2xy    = frame beams with RBS springs at both
ends');
fprintf(fileID, '%s\r\n', '#    6xy    = trusses linking frame and P-delta
column');
fprintf(fileID, '%s\r\n', '#    7xy    = P-delta columns');
fprintf(fileID, '%s\r\n', '#    2xya   = frame beams between panel zone and RBS
spring');
fprintf(fileID, '%s\r\n', '#    3xya   = frame column rotational springs');
fprintf(fileID, '%s\r\n', '#    4xya   = frame beam rotational springs');
fprintf(fileID, '%s\r\n', '#    5xya   = P-delta column rotational springs');
fprintf(fileID, '%s\r\n', '#    4xy00  = panel zone rotational springs');
fprintf(fileID, '%s\r\n', '#    500xya = panel zone elements');
fprintf(fileID, '%s\r\n', '#    where:');
fprintf(fileID, '%s\r\n', '#          x = Pier or Bay #');
fprintf(fileID, '%s\r\n', '#          y = Floor or Story #');
fprintf(fileID, '%s\r\n', '#          a = an integer describing the location
relative to beam-column joint (see description where elements and nodes are
defined)');

fprintf(fileID, '%s\r\n',
'################################################################################
##################','#          Set Up & Source
Definition','################################################################################
################################');

fprintf(fileID, '%s\r\n','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');

fprintf(fileID, '%s\r\n',
'################################################################################
##################','#          Define Analysis
```

```
Type','##############################################################################
########################');

fprintf(fileID, '%s\r\n', 'set dataDir Gravity-Analysis;','file mkdir $dataDir;');

fprintf(fileID, '%s\r\n',
'##############################################################################
##################','#          Define Building, Geometry, Nodes, Masses and
Constraints','##############################################################################
################################');

fprintf(fileID, '%s\r\n', 'set n10;',' ');

fprintf(fileID, '%s%d\r\n', 'set NStories ', NumS);
fprintf(fileID, '%s\r\n', 'set NBays 3');
fprintf(fileID, '%s%d\r\n', 'set WBay ',bay);
fprintf(fileID, '%s%d\r\n', 'set HStory1 ',h1);
fprintf(fileID, '%s%d\r\n', 'set HStoryTyp ',hi);
fprintf(fileID, '%s%d\r\n', 'set HBuilding ',(h1+(NumS-1)*hi));
fprintf(fileID, '%s\r\n', 'source Nodes.tcl','source Masses.tcl');
fprintf(fileID, '%s\r\n', ' ',' # assign boundary condidtions ','# command:  fix
nodeID dxFixity dyFixity rzFixity','# fixity values: 1 = constrained; 0 =
unconstrained','# fix the base of the building; pin P-delta column at base');
fprintf(fileID, '%s\r\n', '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;');
fprintf(fileID, '%s\r\n', '# P-delta column is pinned');
fprintf(fileID, '%s\r\n', 'set transfTag 1;','geomTransf PDelta $transfTag; ', '#
PDelta transformation',' ');
fprintf(fileID, '%s\r\n', 'source Elements.tcl','source Material.tcl','source
ZeroLengthElem.tcl');
fprintf(fileID, '%s\r\n','',
'##############################################################################
##################','#          Gravity Loads & Gravity
Analysis','##############################################################################
##############################');
fprintf(fileID, '%s\r\n', '','# apply gravity loads','#command: pattern
PatternType $PatternID TimeSeriesType','pattern Plain 101 Constant {');
fprintf(fileID, '%s\r\n', '# point loads on leaning column nodes','# command: load
node Fx Fy Mz');
fprintf(fileID, '%s\r\n', 'set P_PD2 [expr -568.00];      # Floor 2','set P_PD3
[expr -565.20];    # Floor 3');
fprintf(fileID, '%s%d', 'set P_PD',NumS+1,' [expr -511.20];     # Floor
',NumS+1');
fprintf(fileID, '%s\r\n','  ',' ');
fprintf(fileID, '%s%d', 'load 5',2,' 0.0 $P_PD',2,' 0.0;  # Floor ',2);
fprintf(fileID, '\n');

k=3;


for i=3:NumS
    fprintf(fileID, '%s%d', 'load 5',k,' 0.0 $P_PD',3,' 0.0;    # Floor ',k);
    fprintf(fileID, '\n');


k=k+1;

end
```

```
fprintf(fileID, '%s%d', 'load 5',NumS+1,' 0.0 $P_PD',NumS+1,' 0.0;           #
Floor ',NumS+1);
    fprintf(fileID, '\n');


fprintf(fileID, '%s\r\n', ' ',' # point loads on frame column nodes',' ');

k=2;

for i=2:NumS
    fprintf(fileID, '%s%d','set P_F',k,'1 [expr -70.21];  # load on each frame
node in Floor ',k,' (exterior)');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','set P_F',k,'2 [expr -46.14];  # load on each frame
node in Floor ',k,' (interior)');
    fprintf(fileID, '\n');
    k=k+1;
end

fprintf(fileID, '%s%d','set P_F',NumS+1,'1 [expr -61.77]; # load on each frame
node in Floor ',NumS+1,' (exterior)');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','set P_F',NumS+1,'2 [expr -41.18];   # load on each
frame node in Floor ',NumS+1,' (interior)');
    fprintf(fileID, '\n');

k=2;

fprintf(fileID, '%s\r\n','  ',' ');

for i=2:NumS+1
    fprintf(fileID, '%s%d','# Floor ',k,' loads');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 1',k,'7 0.0 $P_F',k,'1 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 2',k,'7 0.0 $P_F',k,'2 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 3',k,'7 0.0 $P_F',k,'2 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 4',k,'7 0.0 $P_F',k,'1 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s\r\n','  ',' ');
    k=k+1;
end

fprintf(fileID, '%s\r\n', '','','recorder Element -file
$dataDir/Fcol_static_forces.out -region 4 force;','recorder Node -file
$dataDir/Vbasegravity.out -node 11 21 31 41 51 -dof 1 2 3 reaction;');

fprintf(fileID, '%s\r\n', '','','     }');
fprintf(fileID, '%s\r\n', '','','# Gravity-analysis: load-controlled static
analysis');
fprintf(fileID, '%s\r\n','set Tol 1.0e-6;                       # convergence
tolerance for test');
fprintf(fileID, '%s\r\n','constraints Plain;             # how it handles
boundary conditions');
```

```
fprintf(fileID, '%s\r\n','numberer RCM;                          # renumber dofs
to minimize band-width (optimization)');
fprintf(fileID, '%s\r\n','system BandGeneral;              # how to store and solve
the system of equations in the analysis (large model: try UmfPack)');
fprintf(fileID, '%s\r\n','test NormDispIncr $Tol 6;        # determine if
convergence has been achieved at the end of an iteration step');
fprintf(fileID, '%s\r\n','algorithm Newton;                # use Newtons solution
algorithm: updates tangent stiffness at every iteration');
fprintf(fileID, '%s\r\n','set NstepGravity 1;              # apply gravity in 10
steps');
fprintf(fileID, '%s\r\n','set DGravity [expr 1.0/$NstepGravity]; # load
increment');
fprintf(fileID, '%s\r\n','integrator LoadControl $DGravity;      # determine the
next time step for an analysis');
fprintf(fileID, '%s\r\n','analysis Static;                 # define type of
analysis: static or transient');
fprintf(fileID, '%s\r\n','analyze $NstepGravity;                 # apply
gravity');
fprintf(fileID, '%s\r\n','','# maintain constant gravity loads and reset time to
zero');
fprintf(fileID, '%s\r\n','loadConst -time 0.0');
fprintf(fileID, '%s\r\n','puts "Model Built"','','','');

fprintf(fileID,
'%s\r\n','################################################################################
####','#                        Eigenvalue Analysis

','################################################################################','
','');

fprintf(fileID, '%s\r\n','set pi [expr 2.0*asin(1.0)];                #
Definition of pi');

allLetters='A':'Z';
for i=1:NumS
    aux_letters=['set nEigen',allLetters(i),' ', num2str(i),';'              '
# mode ',allLetters(i), '=', num2str(i)];
    fprintf(fileID,'%s\r\n', aux_letters);
end
fprintf(fileID, '%s\r\n', '');
aux_letters2=['set lambdaNN [eigen  -fullGenLapack  [expr
$nEigen',allLetters(i),']];','   # eigenvalue analysis for nEigen',allLetters(i),'
modes'];
fprintf(fileID, '%s\r\n',aux_letters2);
fprintf(fileID, '%s\r\n', '');

%fprintf(fileID, '%s\r\n','     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','','');
k=1;
for i=1:NumS
```

```
     aux_letters3=['set lambda',allLetters(i),' [lindex $lambdaNN [expr
$nEigen',allLetters(i), '-1]];','   # eigenvalue mode
',allLetters(i),'=',num2str(i)];
     fprintf(fileID, '%s%d',aux_letters3);
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');

k=1;
for i=1:NumS
     aux_letters4=[' [expr pow($lambda',allLetters(i),',0.5)]; # w',num2str(i),
'(',num2str(i), 'st mode circular frequency)'];
     fprintf(fileID, '%s%d','set w',k,aux_letters4);
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');


k=1;
for i=1:NumS

     fprintf(fileID, '%s%d','set T',k,' [expr 2.0*$pi/$w',k,']; # mode period of
the structure');
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');

k=1;

for i=1:NumS

     fprintf(fileID, '%s%d','puts "T',k,' = $T',k,' s";   # display the ',k,' mode
period in the command window');
     fprintf(fileID, '\n');
     k=k+1;
end
```

Dynamic tcl generator

```
function dynamicTclGenerator(numberOfFloors)
    fileID = fopen(['dynamic','.tcl'],'w');


    fprintf(fileID, '\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', '# Start ground motion iteration:');
    fprintf(fileID, '\t%s\r\n', 'set GMinput [open "record.txt" r];');
    fprintf(fileID, '\t%s\r\n', 'set GMs [split [read $GMinput] "\n"]; # each line
contains 4 elements');
    fprintf(fileID, '\t%s\r\n', '#set GmsP [lrange $GMs 0 end-1];');
    fprintf(fileID, '\t%s\r\n', 'close $GMinput;');
    fprintf(fileID, '\t\t%s\r\n', '');
```

```
    fprintf(fileID, '\t%s\r\n', 'foreach GMrecord "$GMs" {;');
    fprintf(fileID, '\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', 'wipe');
    fprintf(fileID, '\t%s\r\n', 'source GravityGeneral.tcl');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', '# display deformed shape:');
    fprintf(fileID, '\t%s\r\n', '#set ViewScale 10;                    # amplify
display of deformed shape');
    fprintf(fileID, '\t%s\r\n', '#DisplayModel2D DeformedShape $ViewScale;    #
display deformed shape, the scaling factor needs to be adjusted for each model');
    fprintf(fileID, '\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', '# Rayleigh Damping');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', '# calculate damping parameters');
    fprintf(fileID, '\t\t%s\r\n', 'set zeta 0.0250;                          #
percentage of critical damping');
    fprintf(fileID, '\t\t%s\r\n', 'set n 10;');
    fprintf(fileID, '\t\t%s\r\n', 'set a0 [expr $zeta*2.0*$w1*$w3/($w1 + $w3)];
      # mass damping coefficient based on first and second modes');
    fprintf(fileID, '\t\t%s\r\n', 'set a1 [expr $zeta*2.0/($w1 + $w3)];
      # stiffness damping coefficient based on first and second modes');
    fprintf(fileID, '\t\t%s\r\n', 'set a1_mod [expr $a1*(1.0+$n)/$n];         #
modified stiffness damping coefficient used for n modified elements. See Zareian &
Medina 2010.');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', '# assign damping to frame beams and columns
            ');
    fprintf(fileID, '\t\t%s\r\n', '# command: region $regionID -eleRange
$elementIDfirst $elementIDlast rayleigh $alpha_mass $alpha_currentStiff
$alpha_initialStiff $alpha_committedStiff');
    fprintf(fileID, '\t\t%s\r\n', '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 rayleigh 0.0 0.0 $a1_mod 0.0;');
    fprintf(fileID, '\t\t\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', 'rayleigh $a0 0.0 0.0 0.0;
                  # assign mass proportional damping to structure (only assigns
to nodes with mass)');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', '# define ground motion parameters');
    fprintf(fileID, '\t\t%s\r\n', 'set patternID 1;       # load pattern ID');
    fprintf(fileID, '\t\t%s\r\n', 'set GMdirection 1;           # ground motion
direction (1 = x)');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', 'set j [lindex [split $GMrecord] 0]; ');
    fprintf(fileID, '\t\t%s\r\n', 'set name [lindex [split $GMrecord] 1];');
    fprintf(fileID, '\t\t%s\r\n', 'set NPTS [lindex [split $GMrecord] 2];');
    fprintf(fileID, '\t\t%s\r\n', 'set deltat [lindex [split $GMrecord] 3];');
    fprintf(fileID, '\t\t%s\r\n', 'set scalefactor [lindex [split $GMrecord]
4];');
    fprintf(fileID, '\t\t%s\r\n', 'set GMtime [expr $NPTS*$deltat + 0.00];    #
total time of ground motion + 10 sec of free vibration');
    fprintf(fileID, '\t\t\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', '# define the acceleration series for the ground
motion');
```

```
    fprintf(fileID, '\t%s\r\n', '# syntax:  "Series -dt $timestep_of_record -
filePath $filename_with_acc_history -factor $scale_record_by_this_amount');
    fprintf(fileID, '\t\t%s\r\n', 'set accelSeries "Series -dt $deltat -filePath
$name -factor [expr $scalefactor*$g]";');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', '# create load pattern: apply acceleration to all
fixed nodes with UniformExcitation');
    fprintf(fileID, '\t\t%s\r\n', '# command: pattern UniformExcitation $patternID
$GMdir -accel $timeSeriesID ');
    fprintf(fileID, '\t\t%s\r\n', 'pattern UniformExcitation $j $GMdirection -
accel $accelSeries;');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', 'set dataDir Results');
    fprintf(fileID, '\t\t%s\r\n', 'file mkdir $dataDir');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\r\n');
    fprintf(fileID, '\t\t%s\r\n', 'recorder Node -file $dataDir/R$j.out -time -
node 11 21 31 41 51 -dof 1 2 3 reaction;');
    stringBuilder = 'recorder Drift -file $dataDir/Drift$j.out -time -iNode 11';
    for i=2:numberOfFloors
        stringBuilder = append(stringBuilder, "1", num2str(i), "005 ");
    end
    stringBuilder = append(stringBuilder, "-jNode ");
    for i=2:numberOfFloors
        stringBuilder = append(stringBuilder, "1", num2str(i), "005 ");
    end
    stringBuilder = append(stringBuilder, '-dof 1 -perpDirn 2;');
    fprintf(fileID, '\t\t%s\r\n', stringBuilder);
    stringBuilder = 'recorder Node -file $dataDir/Accel$j.out -time -node ';
    for i=2:numberOfFloors
        stringBuilder = append(stringBuilder, "1", num2str(i), "005 ");
    end
    stringBuilder = append(stringBuilder, '-dof 1 accel;');
    fprintf(fileID, '\t\t%s\r\n', stringBuilder);
    fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFbase-Connec-
Mom$j.out -time -ele 1 2 3 4 force;');
    fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFbase-Connec-
Rot$j.out -time -ele 1 2 3 4 deformation;');
    fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFcolbase-
Mom$j.out -time -ele 3111 3211 3311 3411 force;');
    fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFcolbase-
Rot$j.out -time -ele 3111 3211 3311 3411 deformation;');

    for i=1:numberOfFloors
        firstLine = append('recorder Element -file $dataDir/MR1col',num2str(i),'-
Mom$j.out -time -ele 31',num2str(i),'1 32',num2str(i),'1 33',num2str(i),'1
34',num2str(i),'1 force;');
        secondLine = append('recorder Element -file $dataDir/MR2col',num2str(i),'-
Mom$j.out -time -ele 31',num2str(i),'2 32',num2str(i),'2 33',num2str(i),'2
34',num2str(i),'2 force;');
        thirdLine = append('recorder Element -file $dataDir/MR1col',num2str(i),'-
Rot$j.out -time -ele 31',num2str(i),'1 32',num2str(i),'1 33',num2str(i),'1
34',num2str(i),'1 deformation;');
        fourthLine = append('recorder Element -file $dataDir/MR2col',num2str(i),'-
Rot$j.out -time -ele 31',num2str(i),'2 32',num2str(i),'2 33',num2str(i),'2
34',num2str(i),'2 deformation;');
        fprintf(fileID, '\t\t%s\r\n', firstLine);
        fprintf(fileID, '\t\t%s\r\n', secondLine);
```

```
        fprintf(fileID, '\t\t%s\r\n', thirdLine);
        fprintf(fileID, '\t\t%s\r\n', fourthLine);
    end

    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRcol1-
Mom$j.out -time -ele 3112 3212 3312 3412 force;');
    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRcol1-
Rot$j.out -time -ele 3112 3212 3312 3412 deformation;');
    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRcol2-
Mom$j.out -time -ele 3121 3221 3321 3421 force;');
    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRcol2-
Rot$j.out -time -ele 3121 3221 3321 3421 deformation;');

    for i=1:numberOfFloors
        firstLine = append('recorder Element -file $dataDir/MRFbeam',num2str(i),'-
Mom$j.out -time -ele 41',num2str(i),'2 42',num2str(i),'1 42', num2str(i),'2 43',
num2str(i),'1 43',num2str(i),'2 44',num2str(i),'1 force;');
        secondLine = append('recorder Element -file
$dataDir/MRFbeam',num2str(i),'-Rot$j.out -time -ele 41',num2str(i),'2
42',num2str(i),'1 42', num2str(i),'2 43', num2str(i),'1 43',num2str(i),'2
44',num2str(i),'1 deformation;');
        fprintf(fileID, '\t\t%s\r\n', firstLine);
        fprintf(fileID, '\t\t%s\r\n', secondLine);
    end

    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFbeam2-
Mom$j.out -time -ele 4122 4221 4222 4321 4322 4421 force;');
    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFbeam2-
Rot$j.out -time -ele 4122 4221 4222 4321 4322 4421 deformation;');
    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFbeam3-
Mom$j.out -time -ele 4132 4231 4232 4331 4332 4431 force;');
    % fprintf(fileID, '\t\t%s\r\n', 'recorder Element -file $dataDir/MRFbeam3-
Rot$j.out -time -ele 4132 4231 4232 4331 4332 4431 deformation;');



    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n', '# define dynamic analysis parameters');
    fprintf(fileID, '\t\t%s\r\n', 'set DtAnalysis 0.001;        # timestep of
analysis');
    fprintf(fileID, '\t\t%s\r\n', 'wipeAnalysis                 # destroy all
components of the Analysis object, i.e. any objects created with system, numberer,
constraints, integrator, algorithm, and analysis commands');
    fprintf(fileID, '\t\t%s\r\n', 'constraints Transformation;  # how it handles
boundary conditions');
    fprintf(fileID, '\t\t%s', 'numberer RCM;              # renumber dof');
    fprintf(fileID, ''''%s\r\n', 's to minimize band-width (optimization)');
    fprintf(fileID, '\t\t%s\r\n', 'system UmfPack;              # how to store
and solve the system of equations in the analysis');
    fprintf(fileID, '\t\t%s\r\n', 'test NormUnbalance 0.01 100; # type of
convergence criteria with tolerance, max iterations');
    fprintf(fileID, '\t\t%s\r\n', 'set algorithmType  Newton; #ModifiedNewton  ');
    fprintf(fileID, '\t\t%s\r\n', 'algorithm $algorithmType;    ');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s', '#algorithm Newton;         # use Newton');
```

```
    fprintf(fileID, '''%s\r\n', 's solution algorithm: updates tangent stiffness
at every iteration');
    fprintf(fileID, '\t\t%s', 'integrator Newmark 0.5 0.25;      # uses Newmark');
    fprintf(fileID, '''%s\r\n', 's average acceleration method to compute the time
history');
    fprintf(fileID, '\t\t%s\r\n', 'analysis Transient;          # type of
analysis: transient or static');
    fprintf(fileID, '\t\t%s\r\n', 'set NumSteps [expr round(($GMtime +
0.0)/$DtAnalysis)];# number of steps in analysis');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t%s\r\n',' puts "";');
    fprintf(fileID, '\t\t%s\r\n', 'puts "Beginning Ground Motion Analysis";');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', 'for {set i 1} {$i <= $NumSteps} {incr i 1} {');
    fprintf(fileID, '\t\t\t%s\r\n', 'set ok [analyze 1 $DtAnalysis];
      # actually perform analysis -- returns ok=0 if analysis was successful');
    fprintf(fileID, '\t\t\t%s\r\n', 'puts "GM step $i out of $NumSteps: $ok";');
    fprintf(fileID, '\t\t%s\r\n', '     if {$ok != 0} {');
    fprintf(fileID, '\t\t\t\t%s\r\n', 'source GM_convergence_loop.tcl;');
    fprintf(fileID, '\t\t\t%s\r\n', '};');
    fprintf(fileID, '\t\t%s\r\n', '}');
    fprintf(fileID, '\t\t%s\r\n', '');
    fprintf(fileID, '\t\t%s\r\n', 'puts "End of Ground Motion Analysis";');
    fprintf(fileID, '\t\r\n');
    fprintf(fileID, '\t%s\r\n', '# output time at end of analysis      ');
    fprintf(fileID, '\t\t%s\r\n', 'set currentTime [getTime];    # get current
analysis time      (after dynamic analysis)');
    fprintf(fileID, '\t\t%s\r\n', 'puts "The current time is: $currentTime";');
    fprintf(fileID, '\t\t\r\n');
    fprintf(fileID, '\t\t\r\n');
    fprintf(fileID, '\t%s\r\n', '}');
end
```

Lateral generator

```
aux_dir2=pwd;
cambio2=[aux_dir2 '\Gravity-Analysis'];
cd(cambio2);

eigen=load ('eigenvalues.out');
cd ..

fileID = fopen(['Lateral','.tcl'],'w');
fprintf(fileID, '%s\r\n', '# Element ID conventions:');
fprintf(fileID, '%s\r\n', '');
fprintf(fileID, '%s\r\n', '#    1xy    = frame columns with RBS springs at both
ends');
fprintf(fileID, '%s\r\n', '#    2xy    = frame beams with RBS springs at both
ends');
fprintf(fileID, '%s\r\n', '#    6xy    = trusses linking frame and P-delta
column');
fprintf(fileID, '%s\r\n', '#    7xy    = P-delta columns');
fprintf(fileID, '%s\r\n', '#    2xya   = frame beams between panel zone and RBS
spring');
fprintf(fileID, '%s\r\n', '#    3xya   = frame column rotational springs');
fprintf(fileID, '%s\r\n', '#    4xya   = frame beam rotational springs');
fprintf(fileID, '%s\r\n', '#    5xya   = P-delta column rotational springs');
```

```
fprintf(fileID, '%s\r\n', '#    4xy00  = panel zone rotational springs');
fprintf(fileID, '%s\r\n', '#    500xya = panel zone elements');
fprintf(fileID, '%s\r\n', '#    where:');
fprintf(fileID, '%s\r\n', '#          x = Pier or Bay #');
fprintf(fileID, '%s\r\n', '#          y = Floor or Story #');
fprintf(fileID, '%s\r\n', '#          a = an integer describing the location
relative to beam-column joint (see description where elements and nodes are
defined)');

fprintf(fileID, '%s\r\n',
'################################################################################
##################','#          Set Up & Source
Definition','################################################################
################################');

fprintf(fileID, '%s\r\n','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');

fprintf(fileID, '%s\r\n',
'################################################################################
##################','#          Define Analysis
Type','################################################################################
########################');

fprintf(fileID, '%s\r\n', '# Define type of analysis:  "pushover" = pushover;
"dynamic" = dynamic');

if typegen==2
    fprintf(fileID, '%s\r\n', 'set analysisType "pushover"');
end
if typegen==3
    fprintf(fileID, '%s\r\n', 'set analysisType "pushover"');
end

if typegen==4
    fprintf(fileID, '%s\r\n', 'set analysisType "dynamic"');
end

fprintf(fileID, '%s\r\n','if {$analysisType == "pushover"} {');
fprintf(fileID, '%s\r\n','   set dataDir Concentrated-PanelZone-Pushover-Output;
      # name of output folder');
fprintf(fileID, '%s\r\n','   file mkdir $dataDir;         # create output
folder');
fprintf(fileID, '%s\r\n','}');
fprintf(fileID, '%s\r\n',
'################################################################################
##################','#          Define Building, Geometry, Nodes, Masses and
Constraints','################################################################
################################');
fprintf(fileID, '%s\r\n', 'set n10;',' ');
fprintf(fileID, '%s%d\r\n', 'set NStories ', NumS);
fprintf(fileID, '%s\r\n', 'set NBays 3');
fprintf(fileID, '%s%d\r\n', 'set WBay ',bay);
fprintf(fileID, '%s%d\r\n', 'set HStory1 ',h1);
```

```
fprintf(fileID, '%s%d\r\n', 'set HStoryTyp ',hi);
fprintf(fileID, '%s%d\r\n', 'set HBuilding ',(h1+(NumS-1)*hi));
fprintf(fileID, '%s\r\n', 'source Nodes.tcl','source Masses.tcl');
fprintf(fileID, '%s\r\n', ' ','# assign boundary condidtions ','# command:  fix
nodeID dxFixity dyFixity rzFixity','# fixity values: 1 = constrained; 0 =
unconstrained','# fix the base of the building; pin P-delta column at base');
fprintf(fileID, '%s\r\n', '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;');
fprintf(fileID, '%s\r\n', '# P-delta column is pinned');
fprintf(fileID, '%s\r\n', 'set transfTag 1;','geomTransf PDelta $transfTag; ', '#
PDelta transformation',' ');

fprintf(fileID, '%s\r\n', 'source Elements.tcl','','#DisplayModel2D
NodeNumbers','source Material.tcl','source ZeroLengthElem.tcl');


fprintf(fileID, '%s\r\n','',
'################################################################################
##################','#          Gravity Loads & Gravity
Analysis','################################################################################
############################');
fprintf(fileID, '%s\r\n', '','# apply gravity loads','#command: pattern
PatternType $PatternID TimeSeriesType','pattern Plain 101 Constant {');
fprintf(fileID, '%s\r\n', '# point loads on leaning column nodes','# command: load
node Fx Fy Mz');
fprintf(fileID, '%s\r\n', 'set P_PD2 [expr -568.00];      # Floor 2','set P_PD3
[expr -565.20];    # Floor 3');
fprintf(fileID, '%s%d', 'set P_PD',NumS+1,' [expr -511.20];     # Floor
',NumS+1');
fprintf(fileID, '%s\r\n',' ',' ');
fprintf(fileID, '%s%d', 'load 5',2,' 0.0 $P_PD',2,' 0.0;  # Floor ',2);
fprintf(fileID, '\n');

k=3;

for i=3:NumS
    fprintf(fileID, '%s%d', 'load 5',k,' 0.0 $P_PD',3,' 0.0;    # Floor ',k);
    fprintf(fileID, '\n');
    k=k+1;
end

fprintf(fileID, '%s%d', 'load 5',NumS+1,' 0.0 $P_PD',NumS+1,' 0.0;           #
Floor ',NumS+1);
fprintf(fileID, '\n');
fprintf(fileID, '%s\r\n', ' ','# point loads on frame column nodes',' ');

k=2;

for i=2:NumS
    fprintf(fileID, '%s%d','set P_F',k,'1 [expr -70.21];  # load on each frame
node in Floor ',k,' (exterior)');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','set P_F',k,'2 [expr -46.14];  # load on each frame
node in Floor ',k,' (interior)');
    fprintf(fileID, '\n');
    k=k+1;
end
```

```
fprintf(fileID, '%s%d','set P_F',NumS+1,'1 [expr -61.77]; # load on each frame
node in Floor ',NumS+1,' (exterior)');
fprintf(fileID, '\n');
fprintf(fileID, '%s%d','set P_F',NumS+1,'2 [expr -41.18]; # load on each frame
node in Floor ',NumS+1,' (interior)');
fprintf(fileID, '\n');

k=2;

fprintf(fileID, '%s\r\n','  ',' ');

for i=2:NumS+1
    fprintf(fileID, '%s%d','# Floor ',k,' loads');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 1',k,'7 0.0 $P_F',k,'1 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 2',k,'7 0.0 $P_F',k,'2 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 3',k,'7 0.0 $P_F',k,'2 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s%d','load 4',k,'7 0.0 $P_F',k,'1 0.0;');
    fprintf(fileID, '\n');
    fprintf(fileID, '%s\r\n','  ',' ');
    k=k+1;
end

fprintf(fileID, '%s\r\n', '','','      }');
fprintf(fileID, '%s\r\n', '','','# Gravity-analysis: load-controlled static
analysis');
fprintf(fileID, '%s\r\n','set Tol 1.0e-6;                        # convergence
tolerance for test');
fprintf(fileID, '%s\r\n','constraints Plain;              # how it handles
boundary conditions');
fprintf(fileID, '%s\r\n','numberer RCM;                          # renumber dofs
to minimize band-width (optimization)');
fprintf(fileID, '%s\r\n','system BandGeneral;             # how to store and solve
the system of equations in the analysis (large model: try UmfPack)');
fprintf(fileID, '%s\r\n','test NormDispIncr $Tol 6;       # determine if
convergence has been achieved at the end of an iteration step');
fprintf(fileID, '%s\r\n','algorithm Newton;               # use Newtons solution
algorithm: updates tangent stiffness at every iteration');
fprintf(fileID, '%s\r\n','set NstepGravity 1;             # apply gravity in 10
steps');
fprintf(fileID, '%s\r\n','set DGravity [expr 1.0/$NstepGravity]; # load
increment');
fprintf(fileID, '%s\r\n','integrator LoadControl $DGravity;     # determine the
next time step for an analysis');
fprintf(fileID, '%s\r\n','analysis Static;                # define type of
analysis: static or transient');
fprintf(fileID, '%s\r\n','analyze $NstepGravity;                 # apply
gravity');
fprintf(fileID, '%s\r\n','','# maintain constant gravity loads and reset time to
zero');
fprintf(fileID, '%s\r\n','loadConst -time 0.0');
fprintf(fileID, '%s\r\n','puts "Model Built"','','','');

fprintf(fileID,
'%s\r\n','################################################################################
```

```
####','#                        Eigenvalue Analysis

','################################################################################','
','');

fprintf(fileID, '%s\r\n','set pi [expr 2.0*asin(1.0)];                 #
Definition of pi');

allLetters='A':'Z';
for i=1:NumS
    aux_letters=['set nEigen',allLetters(i),' ', num2str(i),';'              '
# mode ',allLetters(i), '=', num2str(i)];
    fprintf(fileID,'%s\r\n', aux_letters);
end
fprintf(fileID, '%s\r\n', '');
aux_letters2=['set lambdaNN [eigen  -fullGenLapack  [expr
$nEigen',allLetters(i),']];','   # eigenvalue analysis for nEigen',allLetters(i),'
modes'];
fprintf(fileID, '%s\r\n',aux_letters2);
fprintf(fileID, '%s\r\n', '');

%fprintf(fileID, '%s\r\n','     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','','');
k=1;
for i=1:NumS
     aux_letters3=['set lambda',allLetters(i),' [lindex $lambdaNN [expr
$nEigen',allLetters(i), '-1]];','   # eigenvalue mode
',allLetters(i),'=',num2str(i)];
     fprintf(fileID, '%s%d',aux_letters3);
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');

k=1;
for i=1:NumS
     aux_letters4=[' [expr pow($lambda',allLetters(i),',0.5)]; # w',num2str(i),
'(',num2str(i), 'st mode circular frequency)'];
     fprintf(fileID, '%s%d','set w',k,aux_letters4);
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');


k=1;
for i=1:NumS
```

```
     fprintf(fileID, '%s%d','set T',k,' [expr 2.0*$pi/$w',k,']; # mode period of
the structure');
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','  ',' ','');

k=1;

for i=1:NumS

     fprintf(fileID, '%s%d','puts "T',k,' = $T',k,' s";   # display the ',k,' mode
period in the command window');
     fprintf(fileID, '\n');
     k=k+1;
end


fprintf(fileID, '%s\r\n','',
'################################################################################
##################','#          Analysis
Section','################################################################################
############################');
fprintf(fileID, '%s\r\n','',
'################################################################################
##################','#          Pushover
Analysis','################################################################################
############################');

if typegen==2

fprintf(fileID, '%s\r\n','', 'recorder Element -file
$dataDir/Fcol_story_forces.out -region 4 force');
fprintf(fileID, '%s%d','recorder Node -file $dataDir/Disp_Roof.out -time -node
1',NumS+1,'005 -dof 1 disp;');
fprintf(fileID, '\n');
fprintf(fileID, '%s\r\n','recorder Node -file $dataDir/Rot.out -time -node 11 21
31 41 51 -dof 1 2 3 disp;');
fprintf(fileID, '%s\r\n','recorder Node -file $dataDir/Vbase.out -time -node 11 21
31 41  -dof 1 2 3 reaction;');
end

if typegen==3

fprintf(fileID, '%s\r\n','', 'recorder Element -file
$dataDir/Fcol_story_forces.out -region 4 force');
fprintf(fileID, '%s%d','recorder Node -file $dataDir/Disp_Roof.out -time -node
1',NumS+1,'005 -dof 1 disp;');
fprintf(fileID, '\n');
fprintf(fileID, '%s\r\n','recorder Node -file $dataDir/Rot.out -time -node 11 21
31 41 51 -dof 1 2 3 disp;');
fprintf(fileID, '%s\r\n','recorder Node -file $dataDir/Vbase.out -time -node 11 21
31 41  -dof 1 2 3 reaction;');
end

fprintf(fileID, '%s\r\n','','','if {$analysisType == "pushover"} { ');
fprintf(fileID, '%s\r\n','      puts "Running Pushover..."');
```

```
fprintf(fileID, '%s\r\n','# assign lateral loads and create load pattern:  use
ASCE 7-10 distribution');

k=2;


for i=2:NumS+1

     fprintf(fileID, '%s%d','set lat',k,'  ',eigen(k),';  # force on each beam-
column joint in Floor ',k);
     fprintf(fileID, '\n');
     k=k+1;
end

k=2;

fprintf(fileID, '%s\r\n','','','','    pattern Plain 200 Linear {      ');

for i=2:NumS+1

     fprintf(fileID, '%s%d','load 1',k,'005 $lat',k,' 0.0 0.0;');
     fprintf(fileID, '\n');
     k=k+1;
end

fprintf(fileID, '%s\r\n','','','}');
fprintf(fileID, '%s\r\n','# display deformed shape:');
fprintf(fileID, '%s\r\n','set ViewScale 5;');
fprintf(fileID, '%s\r\n','#DisplayModel2D DeformedShape $ViewScale ;   # display
deformed shape, the scaling factor needs to be adjusted for each model');
fprintf(fileID, '%s\r\n','','# displacement parameters');
fprintf(fileID, '%s%d','set IDctrlNode 1',NumS+1,'005;');
fprintf(fileID, '%s%d','# node where disp is read for disp control');
fprintf(fileID, '%s\r\n','','set IDctrlDOF 1;                    # degree of
freedom read for disp control (1 = x displacement)');
fprintf(fileID, '%s\r\n','set Dmax [expr 0.03*$HBuilding];      # maximum
displacement of pushover: 4% roof drift');
fprintf(fileID, '%s\r\n','set Dincr [expr 0.1];                        #
displacement increment');

fprintf(fileID, '%s\r\n','','# analysis commands');
fprintf(fileID, '%s\r\n','constraints Plain;                    # how it handles
boundary conditions');
fprintf(fileID, '%s\r\n','numberer Plain;                              # renumber
dofs to minimize band-width (optimization)');
fprintf(fileID, '%s\r\n','system UmfPack;                           # how to
store and solve the system of equations in the analysis (large model: try
UmfPack)');
fprintf(fileID, '%s\r\n','set Tol 1.e-4;                               #
Convergence Test: tolerance');
fprintf(fileID, '%s\r\n','set maxNumIter 100;               # Convergence Test:
maximum number of iterations that will be performed before "failure to converge"
is returned');
fprintf(fileID, '%s\r\n','set printFlag 0;');
fprintf(fileID, '%s\r\n','set TestType EnergyIncr;');
fprintf(fileID, '%s\r\n','','#test EnergyIncr  1.0e-5 400;                  #
type of convergence criteria with tolerance, max iterations');
fprintf(fileID, '%s\r\n','#test $TestType $Tol $maxNumIter $printFlag;');
```

```
fprintf(fileID, '%s\r\n','test NormUnbalance 0.01 100;','');

fprintf(fileID, '%s\r\n','','set algorithmType Newton;','','algorithm
$algorithmType;','');
fprintf(fileID, '%s\r\n','#algorithm Newton;                    # use Newtons
solution algorithm: updates tangent stiffness at every iteration');

fprintf(fileID, '%s\r\n','integrator DisplacementControl  $IDctrlNode   $IDctrlDOF
$Dincr;      # use displacement-controlled analysis');
fprintf(fileID, '%s\r\n','analysis Static;                      # define type of
analysis: static for pushover');
fprintf(fileID, '%s\r\n','set Nsteps [expr int($Dmax/$Dincr)];# number of pushover
analysis steps','');

fprintf(fileID, '%s\r\n','for {set i 1} {$i <= $Nsteps} {incr i 1} {');
fprintf(fileID, '%s\r\n','      set ok [analyze 1];             # this will
return zero if no convergence problems were encountered');
fprintf(fileID, '%s\r\n','      puts "PO step $i out of $Nsteps: $ok";');
fprintf(fileID, '%s\r\n','   if {$ok != 0} {');
fprintf(fileID, '%s\r\n','             source PO_convergence_loop.tcl;');
fprintf(fileID, '%s\r\n','      };',' };','','');
fprintf(fileID, '%s\r\n','#set ok [analyze $Nsteps];                  # this will
return zero if no convergence problems were encountered');
fprintf(fileID, '%s\r\n','puts "Pushover complete";             # display this
message in the command window','} ','');

fprintf(fileID,
'%s\r\n','','################################################################################
#######','#   Time History/Dynamic Analysis
#','################################################################################',
'');
fprintf(fileID, '%s\r\n','if {$analysisType == "dynamic"} { ');
fprintf(fileID, '%s\r\n','puts "Running dynamic analysis..."','');
fprintf(fileID, '%s\r\n','source dynamic.tcl','','}','','wipe all;');
```

Generate record file

```
clear all;
clc;
close all;
waning off;


 %This program create the "record.txt" files with the corresponding
 %scale factor:


  files = dir('*.tcl');
  numfiles = length(files);
  scale_factor=xlsread('scale.xls');
  %np=load('num_puntos.txt');
  %deltat=load('dt.txt');
  np=[2999   2999   1999   1999   5590   5590   4531   4531   9992   9992   7807   7807
      4096   4096   4096   4096   5437   5437   6000   6000   2200   2200   11186
      11186  7991   7991   7989   7989   2676   2300   8000   8000   2230   2230   1800
      1800   18000  18000  18000  18000  2800   2800   7269   7269];
```

```
deltat= [0.010   0.010  0.010  0.010  0.010  0.010  0.010  0.010  0.010  0.010
    0.005  0.005  0.010  0.010  0.010  0.010  0.005  0.005  0.005  0.005  0.020
    0.020  0.003  0.003  0.005  0.005  0.005  0.005  0.020  0.020  0.005  0.005
    0.010  0.010  0.020  0.020  0.005  0.005  0.005  0.005  0.010  0.010  0.005
    0.005];

for i=1:length(np)
  dt(i)=deltat(i);
  num_points(i)=np(i);
end


for j=1:numfiles
      name1='record';
      name=[name1 num2str(j+5)];
      fileID = fopen([name,'.txt'],'w');
      for i=1:numfiles
          a=files(i).name;
          b=num2str(num_points(i));
          c=num2str(dt(i));
          d=num2str(scale_factor(i,j));
          fprintf(fileID, '%s %s %s %s %s\r\n', num2str(i),a,b,c,d);
      end
end
```

ANEXO B: CÓDIGO APP DESIGNER

```
classdef Interfaz < matlab.apps.AppBase


% Properties that correspond to app components
properties (Access = public)
NonlinearModelsforSteelMomentFramesAppUIFigure matlab.ui.Figure
TabGroup matlab.ui.container.TabGroup
LinearandNonlinearAnalysisTab matlab.ui.container.Tab
Clearbuildingdrawing matlab.ui.control.Button
RyEditField matlab.ui.control.NumericEditField
RyEditFieldLabel_2 matlab.ui.control.Label
FyksiEditField matlab.ui.control.NumericEditField
FyksiEditFieldLabel matlab.ui.control.Label
SteelTypeDropDown matlab.ui.control.DropDown
SteelTypeDropDownLabel matlab.ui.control.Label
RunAnalysis matlab.ui.control.Button
Typicalfloorheight matlab.ui.control.NumericEditField
TypicalFloorHeightinLabel matlab.ui.control.Label
Firstfloorheight matlab.ui.control.NumericEditField
FirstFloorHeightinLabel matlab.ui.control.Label
Baylength matlab.ui.control.NumericEditField
BayLengthinLabel matlab.ui.control.Label
NumberofStoriesDropDown matlab.ui.control.DropDown
NumberofStoriesDropDownLabel matlab.ui.control.Label
GenerateTCLfilesOpenSees matlab.ui.control.Button
NonlinearAnalysisTypeLabel matlab.ui.control.Label
AnalysisType matlab.ui.control.ListBox
InputData matlab.ui.control.Label
Building matlab.ui.control.UIAxes
PushoverCollapseMechanismTab matlab.ui.container.Tab
ClearAnalysisResultsButton matlab.ui.control.Button
DrawCollapseMechanismButton matlab.ui.control.Button
PushoverCollapseMechanismLabel matlab.ui.control.Label
CollapseMechanismBuildingDrawing matlab.ui.control.UIAxes
end


% Callbacks that handle component events
methods (Access = private)


% Value changed function: SteelTypeDropDown
function SteelTypeDropDownValueChanged(app, event)
value = app.SteelTypeDropDown.Value;
if app.SteelTypeDropDown.Value == "ASTM A36"
app.FyksiEditField.Value = 36;
app.RyEditField.Value = 1.5;
elseif app.SteelTypeDropDown.Value == "ASTM A1043"
app.FyksiEditField.Value = 36;
app.RyEditField.Value = 1.3;
```

```
elseif app.SteelTypeDropDown.Value == "ASTM A992"
app.FyksiEditField.Value = 50;
app.RyEditField.Value = 1.1;
elseif app.SteelTypeDropDown.Value == "ASTM A572 Gr. 50"
app.FyksiEditField.Value = 50;
app.RyEditField.Value = 1.1;
elseif app.SteelTypeDropDown.Value == "ASTM A572 Gr. 55"
app.FyksiEditField.Value = 55;
app.RyEditField.Value = 1.1;
elseif app.SteelTypeDropDown.Value == "ASTM A913 Gr. 50"
app.FyksiEditField.Value = 50;
app.RyEditField.Value = 1.1;
elseif app.SteelTypeDropDown.Value == "ASTM A913 Gr. 60"
app.FyksiEditField.Value = 60;
app.RyEditField.Value = 1.1;
end
end


% Button pushed function: GenerateTCLfilesOpenSees
function GenerateTCLfilesOpenSeesButtonPushed(app, event)
NumS = str2num(app.NumberofStoriesDropDown.Value); % Number of stories of the
building
bay = app.Baylength.Value; % Bay length
h1 = app.Firstfloorheight.Value ; % Height of the first story
hi = app.Typicalfloorheight.Value; % Height of the rest of stories
Ry = app.RyEditField.Value;
Fy = app.FyksiEditField.Value;
%PROPERTIES OF THE PLASTIC HINGES:
%type=2 GRAVITY ANALYSIS!!
%type=1 REDUCTION CAPACITY OF THE COLUMN !!
%Member sizes:
%Input the size of the member for each story from bottom to top
%typegeneral==1- Gravity
%typegeneral==2- Pushover No reduction
%typegeneral==3- Pushover Reduction
%typegeneral==4- Dynamic
typegeneral = app.AnalysisType.Value;


if typegeneral=="Linear Gravity Analysis"
typegen = 1;
type=2;
Runner(NumS, bay, h1, hi, Fy, Ry, type, typegen)
GravityGenerator
GravityGeneralGenerator
end


if typegeneral=="Nonlinear Static Analysis - No Column Reduction"
typegen=2;
type=2;
Runner(NumS, bay, h1, hi, Fy, Ry, type, typegen)
```

```
dynamicTclGenerator(NumS)
LateralGenerator
end
if typegeneral=="Nonlinear Static Analysis - Column Reduction"
typegen=3;
type=1;
Runner(NumS, bay, h1, hi, Fy, Ry, type, typegen)
dynamicTclGenerator(NumS)
LateralGenerator
end
if typegeneral=="Nonlinear Dynamic (Response - History) Analysis"
typegen=4;
type=1;
Runner(NumS, bay, h1, hi, Fy, Ry, type, typegen)
dynamicTclGenerator(NumS)
LateralGenerator
end
Y=[];
j=5;
Y(1)=0;
Y(2)=0;
Y(3)=h1;
Y(4)=h1;


for i=1:(NumS-1)
Y(j)=Y(4)+i*hi;
Y(j+1)=Y(j);
j=j+2;
end
k=1;
for i=1:NumS+1
Xbay1(k)=0;
Xbay1(k+1)=bay;
k=k+2;
end
k=1;
for i=1:NumS+1
Xbay2(k)=bay;
Xbay2(k+1)=2*bay;
k=k+2;
end
k=1;
for i=1:NumS+1
Xbay3(k)=2*bay;
Xbay3(k+1)=3*bay;
k=k+2;
end
Y(2*(NumS+1))=Y(2*(NumS+1)-1);
l=1;
m=1;
for i=1:NumS+1
plot(app.Building,Xbay1(l:l+1),Y(l:l+1),'r');
```

```
plot(app.Building,Xbay2(l:l+1),Y(l:l+1),'r');
plot(app.Building,Xbay3(l:l+1),Y(l:l+1),'r');
hold (app.Building,'on')
l=l+2;
end
Xc1b1=[Xbay1(1), Xbay1(1)];
Xc2b1=[Xbay1(2), Xbay1(2)];
Xc3b2=[Xbay2(2), Xbay2(2)];
Xc4b3=[Xbay3(2), Xbay3(2)];
Yg=[Y(1), Y(2*(NumS+1))];
plot(app.Building,Xc1b1,Yg, 'r');
plot(app.Building,Xc2b1,Yg,'r');
plot(app.Building,Xc3b2,Yg,'r');
plot(app.Building,Xc4b3,Yg,'r');
xlim(app.Building,[-1 3*bay+1]);
ylim(app.Building,[0 h1+(NumS-1)*hi+1]);
hold (app.Building,'off')


end


% Button pushed function: RunAnalysis
function RunAnalysisButtonPushed(app, event)
typegeneral = app.AnalysisType.Value;


if typegeneral=="Linear Gravity Analysis"
typegen = 1;
Status=system('opensees.exe gravity.tcl');
end
if typegeneral=="Nonlinear Static Analysis - No Column Reduction"
typegen=2;
Status=system('opensees.exe lateral.tcl');
aux_dir=pwd;
cambio=[aux_dir '\Concentrated-PanelZone-Pushover-Output'];
cd(cambio);
Data=load('Disp_Roof.out');
Vbase=load('Vbase.out');
% Base_mom=load('MRFbase-Mom.out');
% Base_rot=load('MRFbase-Rot.out');
% Base_momm=Base_mom(:,4);
% Base_rott=-Base_rot(:,4);
%
% Column1_mom=load('MRFcol-Mom-1.out');
% Column1_rot=load('MRFcol-Rot-1.out');
% Column1_momm=Column1_mom(:,4);
% Column1_rott=-Column1_rot(:,2);
m=size(Data);
col=m(1,2);
row=m(1,1);
NumS = str2num(app.NumberofStoriesDropDown.Value); % Number of stories of the
building
```

```
bay = app.Baylength.Value; % Bay length
h1 = app.Firstfloorheight.Value ; % Height of the first story
hi = app.Typicalfloorheight.Value; % Height of the rest of stories
Ry = app.RyEditField.Value;
Fy = app.FyksiEditField.Value;
typegeneral = app.AnalysisType.Value;
num=NumS;
%Height of first floor:
H1=h1;
%Height of typical floor:
Hi=hi;
%Height of the building:
H=H1+(num-1)*Hi;
%Weight of the building:
W1=711.38;
Wi=708;
Wroof=666;
W=W1+Wroof+(num-2)*Wi;
for i=1:row
x(i)=Data(i,2)/H;
y(i)=(Vbase(i,2)+Vbase(i,5)+Vbase(i,8)+Vbase(i,11))/W;
end
% axes(handles.ResponseGraph);
% plot(x,y)
% grid on
% set(gca,'GridLineStyle','-','fontsize',25,'XTickLabel',
num2str(get(gca,'XTick').'))
% set(gca,'YDir','reverse')
% xlabel('Roof Displ. over height (%)','FontName', 'Times New
Roman','FontWeight','bold','fontsize',15);
% ylabel('Normalized Vshear/W','FontName', 'Times New
Roman','FontWeight','bold','fontsize',15);
% title('PUSHOVER CURVE','FontName', 'Times New
Roman','FontWeight','bold','FontSize',15)
%
% cd (aux_dir);
cd ..
end
if typegeneral=="Nonlinear Static Analysis - Column Reduction"
typegen=3;
type=1;
Status=system('opensees.exe lateral.tcl');
aux_dir=pwd;
cambio=[aux_dir '\Concentrated-PanelZone-Pushover-Output'];
cd(cambio);
Data=load('Disp_Roof.out');
Vbase=load('Vbase.out');
% Base_mom=load('MRFbase-Mom.out');
% Base_rot=load('MRFbase-Rot.out');
% Base_momm=Base_mom(:,4);
% Base_rott=-Base_rot(:,4);
%
% Column1_mom=load('MRFcol-Mom-1.out');
```

```
% Column1_rot=load('MRFcol-Rot-1.out');
% Column1_momm=Column1_mom(:,4);
% Column1_rott=-Column1_rot(:,2);
m=size(Data);
col=m(1,2);
row=m(1,1);
%Number of floors:
NumS = str2num(app.NumberofStoriesDropDown.Value); % Number of stories of the
building
bay = app.Baylength.Value; % Bay length
h1 = app.Firstfloorheight.Value ; % Height of the first story
hi = app.Typicalfloorheight.Value; % Height of the rest of stories
Ry = app.RyEditField.Value;
Fy = app.FyksiEditField.Value;
typegeneral = app.AnalysisType.Value;
num=NumS;
%Height of first floor:
H1=h1;
%Height of typical floor:
Hi=hi;
%Height of the building:
H=H1+(num-1)*Hi;
%Weight of the building:
W1=711.38;
Wi=708;
Wroof=666;
W=W1+Wroof+(num-2)*Wi;
for i=1:row
x(i)=Data(i,2)/H;
y(i)=(Vbase(i,2)+Vbase(i,5)+Vbase(i,8)+Vbase(i,11))/W;
end
% axes(handles.ResponseGraph);
% plot(x,y)
% grid on
% set(gca,'GridLineStyle','-','fontsize',25,'XTickLabel',
num2str(get(gca,'XTick').'))
% set(gca,'YDir','reverse')
% xlabel('Roof Displ. over height (%)','FontName', 'Times New
Roman','FontWeight','bold','fontsize',15);
% ylabel('Normalized Vshear/W','FontName', 'Times New
Roman','FontWeight','bold','fontsize',15);
% title('PUSHOVER CURVE','FontName', 'Times New
Roman','FontWeight','bold','FontSize',15)
% cd (aux_dir)
cd ..
end
if typegeneral=="Nonlinear Dynamic (Response - History) Analysis"
typegen=4;
Status=system('opensees.exe lateral.tcl');
end


end
```

```
% Button pushed function: Clearbuildingdrawing
function ClearbuildingdrawingButtonPushed(app, event)
clc
cla(app.Building)
end


% Button pushed function: DrawCollapseMechanismButton
function DrawCollapseMechanismButtonPushed(app, event)
cd('C:\Users\diego\OneDrive\Escritorio\Códigos MATLAB Tesis\Collapse Mechanism')


%Plot collapse mechanism
NumS=8;
matrix=importdata('Nodes.tcl',' ',0);
A=matrix.data;
nodes=A(:,2:3);
Push_Over=load('Disp_Roof.out');
level=2500;
scale=1.0;


%Eight Story Building
for i=1:size(nodes,1)
if nodes(i,2)==165.1
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS+1)*scale;
end
if nodes(i,2)==180
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS+1)*scale;
end
if nodes(i,2)==194.9
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS+1)*scale;
end
if nodes(i,2)==321
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS)*scale;
end
if nodes(i,2)==336
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS)*scale;
end
if nodes(i,2)==351
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS)*scale;
end
if nodes(i,2)==414
nodes(i,1)=nodes(i,1)+(Push_Over(level,NumS)+Push_Over(level,NumS-1))*scale/2;
end
if nodes(i,2)==477
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-1)*scale;
end
if nodes(i,2)==492
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-1)*scale;
end
```

```
if nodes(i,2)==507
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-1)*scale;
end
if nodes(i,2)==634.55
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-2)*scale;
end
if nodes(i,2)==648
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-2)*scale;
end
if nodes(i,2)==661.45
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-2)*scale;
end
if nodes(i,2)==726
nodes(i,1)=nodes(i,1)+(Push_Over(level,NumS-2)+Push_Over(level,NumS-3))*scale/2;
end
if nodes(i,2)==790.55
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-3)*scale;
end
if nodes(i,2)==804
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-3)*scale;
end
if nodes(i,2)==817.45
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-3)*scale;
end
if nodes(i,2)==946.55
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-4)*scale;
end
if nodes(i,2)==960
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-4)*scale;
end
if nodes(i,2)==973.45
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-4)*scale;
end
if nodes(i,2)==1038
nodes(i,1)=nodes(i,1)+(Push_Over(level,NumS-4)+Push_Over(level,NumS-5))*scale/2;
end
if nodes(i,2)==1103.95
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-5)*scale;
end
if nodes(i,2)==1116
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-5)*scale;
end
if nodes(i,2)==1128.05
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-5)*scale;
end
if nodes(i,2)==1261.45
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-6)*scale;
end
if nodes(i,2)==1272
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-6)*scale;
end
if nodes(i,2)==1282.55
nodes(i,1)=nodes(i,1)+Push_Over(level,NumS-6)*scale;
```

```
end
end


%Plastic Hinges:
if level==987
pls_1=load('hinges_1.txt');
elseif level==2500
pls_1=load('hinges_2.txt');
end
%Plot all the nodes:
plot(app.CollapseMechanismBuildingDrawing,nodes(:,1),nodes(:,2),'o','markersize',4
,'color','black')
hold(app.CollapseMechanismBuildingDrawing,'on')
% plot(960,0,'O','MarkerFaceColor','red','MarkerSize',10)
% hold on
if level==987 || level==2500
for i=1:4
plot(app.CollapseMechanismBuildingDrawing,nodes(i,1),nodes(1,2),'O','MarkerFaceCol
or','red','MarkerSize',14)
plot(app.CollapseMechanismBuildingDrawing,nodes(i,1),40,'O','MarkerFaceColor','red
','MarkerSize',14)
end
hold(app.CollapseMechanismBuildingDrawing,'on')
plot(app.CollapseMechanismBuildingDrawing,pls_1(:,1),pls_1(:,2),'O','MarkerFaceCol
or','red','MarkerSize',14)
end
hold(app.CollapseMechanismBuildingDrawing,'off')
%Draw the lines of the columns:
a=0;
xc1=0;
xc2=0;
yc1(1)=0;
yc1(2)=194.9;
yc2(1)=165.1;
yc2(2)=321;
for i=3:NumS+1
yc1(i)=yc1(2)+156.1*(i-2);
yc2(i)=yc1(i)+126.1;
end
k=NumS+1;
kk=NumS;
aa=0;
for j=1:5
line(app.CollapseMechanismBuildingDrawing,[xc1+aa,xc2+aa+Push_Over(level,NumS+1)*s
cale], [yc1(1), yc2(1)],'color','black','LineWidth',2.9)
aa=aa+240;
end
for j=1:5
for i=2:NumS
line(app.CollapseMechanismBuildingDrawing,[xc1+a+Push_Over(level,k)*scale,xc2+a+Pu
sh_Over(level,kk)*scale], [yc1(i), yc2(i)],'color','black','LineWidth',2.9)
k=k-1;
```

```
kk=kk-1;
end
a=a+240;
k=NumS+1;
kk=NumS;
end
%Draw the lines of the beams:
xob1=12.25;
xob2=227.5;
dx=240.25;
for i=1:4
xb1(i)=xob1+dx*(i-1);
xb2(i)=xb1(i)+215.25;
end
yb1(1)=180;
yb2(1)=180;
for i=2:NumS
yb1(i)=yb1(1)+156*(i-1);
yb2(i)=yb1(i);
end
k=NumS+1;
for i=1:NumS
for j=1:4
line(app.CollapseMechanismBuildingDrawing,[xb1(j)+Push_Over(level,k)*scale,xb2(j)+
Push_Over(level,k)*scale], [yb1(i), yb2(i)],'color','black','LineWidth',2.9)
                end
                k=k-1;
            end

            %Draw the lines of the panel zones

            x1p(1)=-12.25;
            x2p(1)=12.25;
            for i=2:4
                x1p(i)=x1p(1)+239.75*(i-1);
                x2p(i)=x2p(1)+240.25*(i-1);
            end

            yp1(1)=165.1;
            yp1(2)=321;
            for i=3:NumS
               yp1(i)=yp1(2)+156*(i-2);
            end

            yp2=yp1;
            k=NumS+1;
            for i=1:NumS
                for j=1:4

line(app.CollapseMechanismBuildingDrawing,[x1p(j)+Push_Over(level,k)*scale,
x2p(j)+Push_Over(level,k)*scale], [yp1(i),
yp2(i)],'color','black','LineWidth',2.9);
                end
```

```
                k=k-1;
            end

            k=NumS+1;
            for i=1:NumS
                for j=1:4

line(app.CollapseMechanismBuildingDrawing,[x1p(j)+Push_Over(level,k)*scale,
x2p(j)+Push_Over(level,k)*scale], [yp1(i)+28.95,
yp2(i)+29],'color','black','LineWidth',2.9);
                end
                k=k-1;
            end

            k=NumS+1;
            for i=1:NumS
                for j=1:4

line(app.CollapseMechanismBuildingDrawing,[x1p(j)+Push_Over(level,k)*scale,
x1p(j)+Push_Over(level,k)*scale], [yp1(i),
yp1(i)+29],'color','black','LineWidth',2.9);
                end
                k=k-1;
            end

            k=NumS+1;
            for i=1:NumS
                for j=1:4

line(app.CollapseMechanismBuildingDrawing,[x1p(j)+24.5+Push_Over(level,k)*scale,
x1p(j)+24.5+Push_Over(level,k)*scale], [yp1(i),
yp1(i)+29],'color','black','LineWidth',2.9);
                end
                k=k-1;
            end


        end


        % Button pushed function: ClearAnalysisResultsButton
        function ClearAnalysisResultsButtonPushed(app, event)
            clc
            cla(app.CollapseMechanismBuildingDrawing)
        end
    end

% Component initialization
methods (Access = private)


% Create UIFigure and components
function createComponents(app)
```

```
% Get the file path for locating images
pathToMLAPP = fileparts(mfilename('fullpath'));


% Create NonlinearModelsforSteelMomentFramesAppUIFigure and hide until all
components are created
app.NonlinearModelsforSteelMomentFramesAppUIFigure = uifigure('Visible', 'off');
app.NonlinearModelsforSteelMomentFramesAppUIFigure.Color = [0.9412 0.9412 0.9412];
app.NonlinearModelsforSteelMomentFramesAppUIFigure.Position = [100 100 1059 527];
app.NonlinearModelsforSteelMomentFramesAppUIFigure.Name = 'Nonlinear Models for
Steel Moment Frames App';
app.NonlinearModelsforSteelMomentFramesAppUIFigure.Icon = fullfile(pathToMLAPP,
'Imagen aplicación App Designer.png');


% Create TabGroup
app.TabGroup = uitabgroup(app.NonlinearModelsforSteelMomentFramesAppUIFigure);
app.TabGroup.Position = [1 13 1044 514];


% Create LinearandNonlinearAnalysisTab
app.LinearandNonlinearAnalysisTab = uitab(app.TabGroup);
app.LinearandNonlinearAnalysisTab.Title = 'Linear and Nonlinear Analysis';


% Create Building
app.Building = uiaxes(app.LinearandNonlinearAnalysisTab);
title(app.Building, 'Structure Elevation View')
xlabel(app.Building, 'X (in)')
ylabel(app.Building, 'Y (in)')
zlabel(app.Building, 'Z')
app.Building.XColor = [0 0 0];
app.Building.YColor = [0 0 0];
app.Building.ZColor = [0 0 0];
app.Building.XGrid = 'on';
app.Building.YGrid = 'on';
app.Building.ColorOrder = [0 0 1];
app.Building.TitleFontWeight = 'normal';
app.Building.Position = [532 28 475 423];


% Create InputData
app.InputData = uilabel(app.LinearandNonlinearAnalysisTab);
app.InputData.FontSize = 16;
app.InputData.FontWeight = 'bold';
app.InputData.Position = [84 429 83 22];
app.InputData.Text = 'Input Data';


% Create AnalysisType
app.AnalysisType = uilistbox(app.LinearandNonlinearAnalysisTab);
```

```
app.AnalysisType.Items = {'Linear Gravity Analysis', 'Nonlinear Static Analysis -
No Column Reduction', 'Nonlinear Static Analysis - Column Reduction', 'Nonlinear
Dynamic (Response - History) Analysis'};
app.AnalysisType.Position = [184 138 295 77];
app.AnalysisType.Value = 'Linear Gravity Analysis';


% Create NonlinearAnalysisTypeLabel
app.NonlinearAnalysisTypeLabel = uilabel(app.LinearandNonlinearAnalysisTab);
app.NonlinearAnalysisTypeLabel.HorizontalAlignment = 'right';
app.NonlinearAnalysisTypeLabel.Position = [118 167 51 22];
app.NonlinearAnalysisTypeLabel.Text = 'Methods';


% Create GenerateTCLfilesOpenSees
app.GenerateTCLfilesOpenSees = uibutton(app.LinearandNonlinearAnalysisTab,
'push');
app.GenerateTCLfilesOpenSees.ButtonPushedFcn = createCallbackFcn(app,
@GenerateTCLfilesOpenSeesButtonPushed, true);
app.GenerateTCLfilesOpenSees.IconAlignment = 'center';
app.GenerateTCLfilesOpenSees.BackgroundColor = [0.9412 0.9412 0.9412];
app.GenerateTCLfilesOpenSees.FontWeight = 'bold';
app.GenerateTCLfilesOpenSees.Position = [184 83 295 38];
app.GenerateTCLfilesOpenSees.Text = 'Generate TCL Files - OpenSees';


% Create NumberofStoriesDropDownLabel
app.NumberofStoriesDropDownLabel = uilabel(app.LinearandNonlinearAnalysisTab);
app.NumberofStoriesDropDownLabel.HorizontalAlignment = 'right';
app.NumberofStoriesDropDownLabel.Position = [65 387 102 22];
app.NumberofStoriesDropDownLabel.Text = 'Number of Stories';


% Create NumberofStoriesDropDown
app.NumberofStoriesDropDown = uidropdown(app.LinearandNonlinearAnalysisTab);
app.NumberofStoriesDropDown.Items = {'2', '4', '8', '12', '20'};
app.NumberofStoriesDropDown.Editable = 'on';
app.NumberofStoriesDropDown.BackgroundColor = [1 1 1];
app.NumberofStoriesDropDown.Position = [184 382 50 30];
app.NumberofStoriesDropDown.Value = '2';


% Create BayLengthinLabel
app.BayLengthinLabel = uilabel(app.LinearandNonlinearAnalysisTab);
app.BayLengthinLabel.HorizontalAlignment = 'right';
app.BayLengthinLabel.Position = [80 336 87 22];
app.BayLengthinLabel.Text = 'Bay Length (in)';


% Create Baylength
app.Baylength = uieditfield(app.LinearandNonlinearAnalysisTab, 'numeric');
app.Baylength.HorizontalAlignment = 'left';
app.Baylength.Position = [184 332 49 30];
```

```
% Create FirstFloorHeightinLabel
app.FirstFloorHeightinLabel = uilabel(app.LinearandNonlinearAnalysisTab);
app.FirstFloorHeightinLabel.HorizontalAlignment = 'right';
app.FirstFloorHeightinLabel.Position = [48 286 118 22];
app.FirstFloorHeightinLabel.Text = 'First Floor Height (in)';


% Create Firstfloorheight
app.Firstfloorheight = uieditfield(app.LinearandNonlinearAnalysisTab, 'numeric');
app.Firstfloorheight.HorizontalAlignment = 'left';
app.Firstfloorheight.Position = [184 282 50 30];


% Create TypicalFloorHeightinLabel
app.TypicalFloorHeightinLabel = uilabel(app.LinearandNonlinearAnalysisTab);
app.TypicalFloorHeightinLabel.HorizontalAlignment = 'right';
app.TypicalFloorHeightinLabel.Position = [34 238 132 22];
app.TypicalFloorHeightinLabel.Text = 'Typical Floor Height (in)';


% Create Typicalfloorheight
app.Typicalfloorheight = uieditfield(app.LinearandNonlinearAnalysisTab,
'numeric');
app.Typicalfloorheight.HorizontalAlignment = 'left';
app.Typicalfloorheight.Position = [184 233 50 30];


% Create RunAnalysis
app.RunAnalysis = uibutton(app.LinearandNonlinearAnalysisTab, 'push');
app.RunAnalysis.ButtonPushedFcn = createCallbackFcn(app, @RunAnalysisButtonPushed,
true);
app.RunAnalysis.IconAlignment = 'center';
app.RunAnalysis.FontWeight = 'bold';
app.RunAnalysis.Position = [184 29 136 38];
app.RunAnalysis.Text = 'Run Analysis';


% Create SteelTypeDropDownLabel
app.SteelTypeDropDownLabel = uilabel(app.LinearandNonlinearAnalysisTab);
app.SteelTypeDropDownLabel.HorizontalAlignment = 'right';
app.SteelTypeDropDownLabel.Position = [275 383 62 31];
app.SteelTypeDropDownLabel.Text = 'Steel Type';


% Create SteelTypeDropDown
app.SteelTypeDropDown = uidropdown(app.LinearandNonlinearAnalysisTab);
app.SteelTypeDropDown.Items = {'ASTM A36', 'ASTM A1043', 'ASTM A992', 'ASTM A572
Gr. 50', 'ASTM A572 Gr. 55', 'ASTM A913 Gr. 50', 'ASTM A913 Gr. 60'};
app.SteelTypeDropDown.ValueChangedFcn = createCallbackFcn(app,
@SteelTypeDropDownValueChanged, true);
app.SteelTypeDropDown.Position = [344 383 135 31];
```

```
            app.SteelTypeDropDown.Value = 'ASTM A36';


            % Create FyksiEditFieldLabel
            app.FyksiEditFieldLabel = uilabel(app.LinearandNonlinearAnalysisTab);
            app.FyksiEditFieldLabel.HorizontalAlignment = 'right';
            app.FyksiEditFieldLabel.Position = [284 332 45 30];
            app.FyksiEditFieldLabel.Text = 'Fy (ksi)';


            % Create FyksiEditField
            app.FyksiEditField = uieditfield(app.LinearandNonlinearAnalysisTab,
'numeric');
            app.FyksiEditField.HorizontalAlignment = 'left';
            app.FyksiEditField.Position = [345 332 51 30];


            % Create RyEditFieldLabel_2
            app.RyEditFieldLabel_2 = uilabel(app.LinearandNonlinearAnalysisTab);
            app.RyEditFieldLabel_2.HorizontalAlignment = 'right';
            app.RyEditFieldLabel_2.Position = [307 282 25 29];
            app.RyEditFieldLabel_2.Text = 'Ry';


            % Create RyEditField
            app.RyEditField = uieditfield(app.LinearandNonlinearAnalysisTab,
'numeric');
            app.RyEditField.HorizontalAlignment = 'left';
            app.RyEditField.Position = [345 282 51 30];


            % Create Clearbuildingdrawing
            app.Clearbuildingdrawing = uibutton(app.LinearandNonlinearAnalysisTab,
'push');
            app.Clearbuildingdrawing.ButtonPushedFcn = createCallbackFcn(app,
@ClearbuildingdrawingButtonPushed, true);
            app.Clearbuildingdrawing.IconAlignment = 'center';
            app.Clearbuildingdrawing.FontWeight = 'bold';
            app.Clearbuildingdrawing.Position = [342 29 137 38];
            app.Clearbuildingdrawing.Text = 'Reset Analysis';


            % Create PushoverCollapseMechanismTab
            app.PushoverCollapseMechanismTab = uitab(app.TabGroup);
            app.PushoverCollapseMechanismTab.Title = 'Pushover Collapse
Mechanism';


            % Create CollapseMechanismBuildingDrawing
            app.CollapseMechanismBuildingDrawing =
uiaxes(app.PushoverCollapseMechanismTab);
            title(app.CollapseMechanismBuildingDrawing, 'Building with Plastic
Hinges')
```

```
            xlabel(app.CollapseMechanismBuildingDrawing, 'X (in)')
            ylabel(app.CollapseMechanismBuildingDrawing, 'Y (in)')
            zlabel(app.CollapseMechanismBuildingDrawing, 'Z')
            app.CollapseMechanismBuildingDrawing.XGrid = 'on';
            app.CollapseMechanismBuildingDrawing.YGrid = 'on';
            app.CollapseMechanismBuildingDrawing.TitleFontWeight = 'normal';
            app.CollapseMechanismBuildingDrawing.Position = [284 10 731 458];


            % Create PushoverCollapseMechanismLabel
            app.PushoverCollapseMechanismLabel =
uilabel(app.PushoverCollapseMechanismTab);
            app.PushoverCollapseMechanismLabel.HorizontalAlignment = 'center';
            app.PushoverCollapseMechanismLabel.FontSize = 16;
            app.PushoverCollapseMechanismLabel.FontWeight = 'bold';
            app.PushoverCollapseMechanismLabel.Position = [23 292 253 41];
            app.PushoverCollapseMechanismLabel.Text = 'Pushover Collapse
Mechanism';


            % Create DrawCollapseMechanismButton
            app.DrawCollapseMechanismButton =
uibutton(app.PushoverCollapseMechanismTab, 'push');
            app.DrawCollapseMechanismButton.ButtonPushedFcn =
createCallbackFcn(app, @DrawCollapseMechanismButtonPushed, true);
            app.DrawCollapseMechanismButton.Position = [59 233 183 38];
            app.DrawCollapseMechanismButton.Text = 'Draw Collapse Mechanism';


            % Create ClearAnalysisResultsButton
            app.ClearAnalysisResultsButton =
uibutton(app.PushoverCollapseMechanismTab, 'push');
            app.ClearAnalysisResultsButton.ButtonPushedFcn =
createCallbackFcn(app, @ClearAnalysisResultsButtonPushed, true);
            app.ClearAnalysisResultsButton.Position = [60 177 183 38];
            app.ClearAnalysisResultsButton.Text = 'Clear Analysis Results';


            % Show the figure after all components are created
            app.NonlinearModelsforSteelMomentFramesAppUIFigure.Visible = 'on';
        end
    end


    % App creation and deletion
    methods (Access = public)


        % Construct app
        function app = Interfaz


            % Create UIFigure and components
```

```
            createComponents(app)


            % Register the app with App Designer
            registerApp(app, app.NonlinearModelsforSteelMomentFramesAppUIFigure)


            if nargout == 0
                clear app
            end
        end


        % Code that executes before app deletion
        function delete(app)


            % Delete UIFigure when app is deleted
            delete(app.NonlinearModelsforSteelMomentFramesAppUIFigure)
        end
    end
end
```