UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Desarrollo de código MATLAB para interactuar con OpenSees

Pedro José Hidalgo Ortega

Ingeniería Civil

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

Quito, 03 de mayo de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Desarrollo de código MATLAB para interactuar con OpenSees

Pedro José Hidalgo Ortega

Nombre del profesor, Título académico	**Pablo Andrés Torres Rodas, Ing. Civil**

Quito, 03 de mayo de 2021

© DERECHOS DE AUTOR

Nombres y apellidos: Pedro José Hidalgo Ortega

Código: 00136370

Cédula de identidad: 1721489928

Lugar y fecha: Quito, 03 de mayo de 2021

ACLARACIÓN PARA PUBLICACIÓN

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

UNPUBLISHED DOCUMENT

Note: The following 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

Complementario a la propia dificultad que realizar los procesos de análisis estructurales implica, el tiempo y complejidad para definir las condiciones para su análisis son factores incidentes a la hora de decidir sobre el uso de los métodos de análisis y diseño. Aunque estos factores se han reducido de manera considerable gracias a los avances notables en las computadoras como programas que juntan de mejor manera los procesos requeridos, siguen consistiendo en mecanismos tediosos de realización. Por esto, el objetivo de este trabajo es la realización un programa en Excel para automatizar lo más posible el diseño de los elementos correspondientes a un marco arriostrado concéntricamente; así como una interfaz en MATLAB que ejecute y sirva como nexo para la realización de los métodos de análisis no lineal en estructuras mediante la interacción del software OpenSees. Con lo cual, se pudo generar una reducción importante en tiempo requerido para los dos procesos, así como la facilidad de generación de datos necesarios para los análisis requeridos; pero que a la final, representan un punto de partida para poder seguir automatizando estos procesos para volverlos mucho más amigables con el usuario y que permitan obtener resultados más simples de interpretar.

Palabras Clave: CBF, Análisis Plástico, Análisis No-Lineal, Pushover, Tiempo-Historia, Dinámico Incremental, Opensees, MATLAB.

ABSTRACT

Complementary to the own difficulty that carrying out the structural analysis processes implies, the time and complexity to define the conditions for their analysis are incident factors when deciding on the use of the analysis and design methods. Although these factors have been reduced considerably thanks to the notable advances in computers as programs that better bring together the required processes, they still consist of tedious implementation mechanisms. For this reason, the objective of this work is to create an Excel program to automate as much as possible the design of the elements corresponding to a concentrically braced frame; as well as a MATLAB interface that executes and serves as a link to carry out nonlinear analysis methods on structures through the interaction of the OpenSees software. With which, it was possible to generate an important reduction in the time required for the two processes, as well as the ease of generating the data necessary for the required analyzes; but that in the end they represent a starting point to be able to continue automating these processes to make them much more user-friendly and to obtain results that are simpler to interpret.

Key Words: CBF, Plastic Analysis, Non-Linear Analysis, Pushover, Time-History, Dynamic Incremental, Opensees, MATLAB.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

El propósito principal de este trabajo es la elaboración de programas de Excel y MATLAB para facilitar y agilizar el proceso de análisis de datos como la generación de estos para estructuras de acero de tipo arriostradas o marcos CBF. Lo cual permite agilizar y poder post - procesar de manera más limpia los resultados obtenidos en el análisis no lineal de las estructuras en cuestión.

Esto se basa en los procedimientos estándar para el análisis académico de los marcos arriostrados y que son complementados con la interacción con una interfaz especializada en el análisis no lineal de estructuras desarrollado por Pacific Earthquake Engineering Research Center en colaboración con la Universidad de Berkeley denominado OpenSees.

La importancia del trabajo recae principalmente en la necesidad y deseo de optimizar como automatizar el proceso de implementación del programa Opensees para generar una interfaz más amigable con los usuarios y con la posibilidad de generar automáticamente todos los archivos necesarios desde los comandos básicos de una interfaz de MATLAB. Con el fin de procesar, post – procesar y analizar los resultados de los análisis de las estructuras de acero para poder ser adaptados los resultados para el interés de análisis especifico de cada investigador que se vea en la necesidad de realizar este tipo de análisis.

Con relación a la realidad ecuatoriana y en el mundo en general el tema del análisis no lineal de los elementos es un mecanismo de análisis que ha ido tomando importancia para determinar de forma más adecuada el comportamiento de los diversos elementos de una estructura; pero que, debido a su mayor complejidad con respecto a un análisis tradicional, en el Ecuador no se tiende mucho a realizar este tipo de análisis a

menos que sea estrictamente necesario. Es por esto que la importancia de la generación de la interfaz por medio de MATLAB permite agilitar y facilitar el proceso de análisis volviéndolo más amigable con el usuario y que permite obtener una mejor noción de los resultados arrojados post análisis para poder obtener conclusiones más certeras sobre los comportamientos de los elementos interesados en función del tiempo principalmente.

Consecuentemente, es primordial el entender toda la teoría perteneciente al diseño de los marcos arriostrados, así como la filosofía del análisis no lineal de los elementos para poder entender como los códigos fueron desarrollados con las variables necesarias para poder procesar los resultados.

Desarrollo del Tema

Capítulo 1: Generalidades

1.1Estructuras de Acero

Históricamente, las estructuras a lo largo del planeta se han caracterizado por ir desarrollando nuevas técnicas y formas de mejorar sus diseños para acoplarse ante las necesidades venideras. En donde uno de los temas que ha tomado importancia y relevancia es la elaboración de las estructuras con materiales alternativos y complementarios al tradicional concreto; siendo este el caso del acero.

El acero como un material estructural cuenta con algunas ventas notables con respecto a su alternativa clásica del concreto, que a simple vista mostraría al acero como una opción más que razonable. Hablando de la resistencia de los materiales el acero, estos cuentan con un alta hasta muy alta resistencia en comparación con la capacidad del hormigón; pero con la particularidad que el acero posee un peso unitario mucho menor que el concreto. Esto se ve reflejado en un peso total de una estructura mucho menor, que a su vez en el análisis de fuerzas sísmicas para su posterior diseño va a generar fuerzas mucho menores por lo cual la estructura se vuelve más segura y a la larga económica. De igual forma, el hecho de ser un material con menor peso va a ser óptimo para cuando se encuentra en el diseño de estructuras que se salen un poco de las condiciones óptimas; tales como el uso de vanos de largas dimensiones, estructuras de alturas considerables, o hasta en la construcción de estructuras que se encuentran cimentadas sobre suelos con condiciones muy desfavorables. (McCormac & Csernak, 2012)

Complementariamente, es primordial mencionar que el acero estructural cuenta con una uniformidad a lo largo del material lo cual lo vuelve en una estructura estable a través del tiempo en cuanto a sus propiedades; a diferencia del concreto que es un

material completamente heterogéneo por todos sus componentes y su forma de elaboración. Agregando a lo mencionado el hecho que el acero es un material elástico que más se asemeja a las suposiciones tomadas en cuenta en la ley de Hooke; y el hecho que el acero si se lo mantiene de una manera óptima, las estructuras de acero pueden tener una duración prácticamente indefinida.

Otra propiedad del acero estructural que es una ventaja a la hora de la construcción de estructuras esencialmente diseñadas con fines sísmicos es la ductilidad del material. Esta propiedad es fundamental cuando se habla acerca de la seguridad en las estructuras, puesto que es la propiedad del acero que permite la deformación extensa de los elementos sin que estos entren en un estado de falla del material. Esta propiedad vuelve al acero en una opción más que interesante puesto que al tener ductilidades mucho mayores que en el concreto en eventos sísmicos similares a los de diseño representa mejores condiciones de serviciabilidad.

Del otro lado, cabe recalcar que, si bien el acero es un material con una serie de ventaja con respecto al concreto, existen ciertos problemas que pueden generarse en las estructuras que si no se los toma en cuenta pueden ser catastróficos. En primer lugar, debido a la alta resistencia del material, los elementos de acero tienen una susceptibilidad importante al pandeo. Adicionalmente, en las estructuras de acero puede darse el caso de fatiga en los elementos al verse sometido a cargas cíclicas, lo cual puede afectar a las estimaciones de los esfuerzos esperados por los elementos. (McCormac & Csernak, 2012)

1.2 Acero Estructural

En la actualidad, debido a los constantes avances en la tecnología en el ámbito de la construcción, el uso del acero estructural ha ido variando en cierto punto con respecto a

la variedad de aceros disponibles en el mercado; esto no ha representado un cambio considerable en las condiciones de aceptación. Puesto que a pesar de irse generando un catálogo mayor de opciones de acero estructural a implementar; es primordial mostrar que los requerimientos mínimos de aceptación entre las variantes se han mantenido prácticamente igual de simples. Limitando de manera leve la composición química con la que se realiza cada varilla, estableciendo parámetros mínimos de propiedades mecánicas del material tales como el esfuerzo de fluencia o también los porcentajes mínimos de elongación de deformación antes de llegar al estado de falla del elemento. (Bruneau, Chia-Ming, & Sabelli, 2011)

En base a lo mencionado en el párrafo anterior, toma relevancia el papel del ingeniero a la hora de elegir entre las opciones existentes porque al tener una gran cantidad de opciones que cumplen los requerimientos mínimos, es primordial es saber elegir el tipo de acero estructural óptimo en función de las condiciones necesarias para cada proyecto. Siendo fundamental el tener en cuenta la respuesta de la estructura en cuanto a la ductilidad, así como su rendimiento ante las condiciones principalmente ambientales en las cuales se va a encontrar expuesta la estructura. Para lo cual va a ser indispensable que se dé un entendimiento por parte del ingeniero de temas como el comportamiento del acero, su curva de esfuerzos deformaciones y la dependencia que la ductilidad de una estructura va a estar ligada con los esfuerzos que resisten los distintos tipos de acero estructural.

Cabe resaltar, que uno de los temas que se debe tener en cuenta a su vez por su importancia en el diseño sísmico es el esfuerzo de fluencia esperado. Pues el conocimiento del límite elástico máximo probable es tan importante como el conocimiento del límite elástico mínimo confiable. Este tema cobro importancia en el último tiempo porque se ha evidenciado que el margen entre el límite elástico promedio

real y el límite elástico especificado ha aumentado progresivamente a lo largo de los años en algunos tipos aceros estructurales; sin que se generen cambios en las especificaciones de los mismo. Y si bien a simple vista esto puede parecer no afectar al diseño por tener una mayor resistencia, hablando del diseño sísmico, un límite elástico inesperadamente más alto puede ser desventajoso; ya que se puede dar el caso en que elementos diseñado para ceder y disipar energía durante un terremoto por tener un límite mayor no termine fallando, generando una sobrecarga de los esfuerzos generados en los demás elementos estructurales que puede desencadenar en cambios drásticas sobre el comportamiento final de la estructura al punto de poder conducir a una falla frágil en el mecanismo. (Bruneau, Chia-Ming, & Sabelli, 2011)

1.3Marcos Arriostrados Concéntricos CBF

Los marcos arriostrados concéntricamente o denominados CBF por sus siglas en inglés (SCBF) se han convertido en alternativas más populares como sistemas de resistencia de carga lateral en el último tiempo para regiones en donde se consideran con alta actividad sísmica. Tendencia la cual surgió ante las dudas generadas en la sociedad de ingenieros con respecto a la implementación de los marcos especiales resistentes a momento o SMRF. Los marcos resistentes a momentos especiales se encontraban entre los más populares sistemas para soportar cargas laterales, y su implementación era muy amplia hasta los sucesos de los terremotos de Northridge en 1994 y Hyogoken Naubu en 1995; donde se produjeron daños severos en muchas de estas estructuras.

Los marcos arriostrados CBF están diseñados elásticamente como un sistema de pórtico vertical con la capacidad de resistir las cargas laterales a través de los elementos de arriostramiento axiales. Donde, la filosofía de diseño tiene un enfoque orientado

hacia el diseño de capacidad; lo que ha representado en la incorporación por completo de las disposiciones sísmicas más recientes correspondiente para el diseño de edificios de acero estructural. Los CBF disipan la energía del terremoto mediante el pandeo de los tirantes de compresión y la deformación de los tirantes sometidos a tracción. De acuerdo con el enfoque de diseño actual implementado en AISC 341-10, las vigas y columnas en SCBF se diseñarán en función de la capacidad de las riostras para mantener las vigas y columnas en la región elástica. Se agrega una nueva sección de análisis en AISC 341-10 para abordar las respuestas inelásticas de las SCBF. Se requieren dos análisis estructurales separados y un análisis adicional para SCBF en AISC 341-10. Estos requisitos de análisis aumentan significativamente los esfuerzos de diseño en las oficinas de diseño típicas, y parece estar justificado un estudio integral para demostrar cómo un procedimiento de diseño tan explícito e inelástico mejoraría o no significativamente el rendimiento sísmico de los SCBF. (Wen, Seker, Akbas, & Shen, 2016)

Los marcos arriostrados cuentan con diversas configuraciones con relación a la disposición de las riostras en el marco de la estructura. Cabe recalcar que inicialmente las configuraciones de las riostras en las estructuras satisfacían inicialmente los requerimientos para soportar las cargas de viento para las cuales eran diseñadas únicamente; pero con todo el tema sísmico muchas de ellas no son aptas para ser empleadas en el diseño sísmico de alguna estructura. Algunas configuraciones están prohibidas en regiones sísmicas porque exhiben una respuesta inelástica cíclica pobre o inducen demandas indeseables en otros elementos estructurales. Lo cual se encuentra representado de una mejor manera mediante la siguiente imagen en donde se evidencia de manera gráfica las configuraciones principales aceptadas así también como las que no están permitidas.

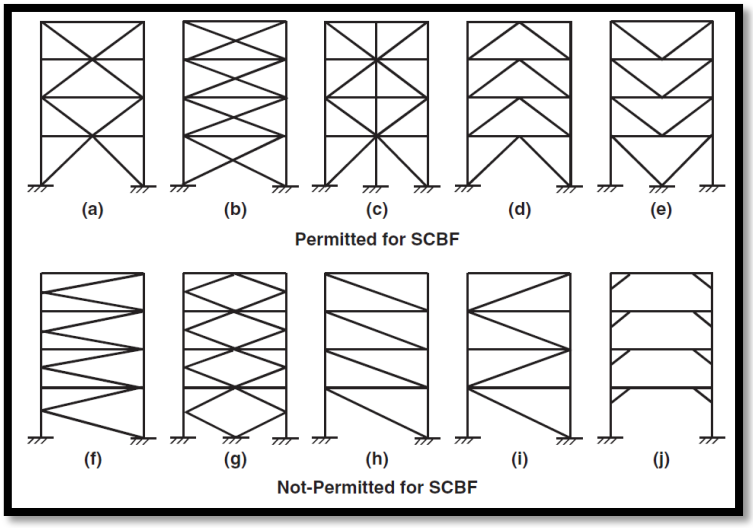


Ilustración 1:Tipos de configuraciones permitidas y no permitidas del uso de riostras en estructuras de acero.

Fuente: (Bruneau, Chia-Ming, & Sabelli, 2011)

La razón principal por la cual existen ciertas configuraciones que se encuentran prohibidas para los marcos especiales es el hecho que, en función de su configuración no simétrica con relación a las fuerzas cíclicas, se puede dar el caso de que una de las riostras diagonales se doblara y provocaría que se dé un fuerte aumento en la fuerza de tracción que sufriría la riostra la cual a su vez se transferiría como cortante en la columna adyacente. Provocando que la fuerza horizontal resultante por el hecho de tener riostras desiguales produzca una bisagra plástica en la columna en el punto de intersección de la riostra y la columna, dando como resultado una falla en la columna lo que se podría traducir en la formación del mecanismo en la estructura y su posterior destrucción. (Bruneau, Chia-Ming, & Sabelli, 2011)

Tal como se puede observar en la figura 1, los arriostramientos tipo V, tipo V invertido (o también denominadas chevron) y X son tres categorías principales de configuraciones de arriostramiento para marcos SCBF. Sin embargo, es importante mencionar que, en el último tiempo en función de estudios realizados, las configuraciones en X han ido perdiendo espacio debido a sus costos de conexión

adicionales, conjuntamente con ciertos estudios donde se ha evidenciado que los marcos arriostrados en X no siempre permanecen elásticos durante un suceso sísmico; yendo en contra de la filosofía de diseño principal para este tipo de estructuras violando el código de diseño actual. El comportamiento inelástico en las vigas se debe al comportamiento cíclico asimétrico de los tirantes que crea una fuerza desequilibrada considerable. Durante las dos últimas décadas, aunque se han realizado extensos estudios realizado sobre el comportamiento sísmico de los SCBF, con la mayoría de estos estudios enfocados en el comportamiento de los arriostramientos, las investigaciones sobre las vigas y pilares han sido dispersas y escasas, aunque se ha estudiado el desempeño de arriostramientos, vigas y pilares que se interrelacionan con unos a otros como sistemas únicos parece necesario e importante. (Momenzadeh, 2017)

Finalmente, tomando en cuenta todos los cambios que se han generado a partir de los hechos suscitados en1994 y 1995 en donde se han establecido requisitos y restricciones más estrictos para el diseño de los marcos SMRF; esto se ve reflejado en vigas y columnas más pesadas en estos sistemas, lo que resulta en detalles de las conexiones con una mayor complejidad y traducidos en altos costos de mano de obra. Por lo cual, los marcos especiales arriostrados concéntricamente SCBF se convierten en una alternativa más ventajosa debido a su relativa facilidad de construcción con relación a los SMRF y que los vuelve más rentables a la vez.

Los marcos arriostrados CBF resultan ser un sistema más económico que permite conseguir resistencia ante cargas laterales a las estructuras basada en la mayor rigidez lateral aportadas por las riostras a la estructura, por lo cual su ocupación se ha convertido en la configuración más habitual para fines sísmicos. La incidencia de su aporte y comportamiento depende principalmente del comportamiento axial de sus arriostramientos. Su comportamiento axial es de carácter asimétrico al tener 2 riostras

con orientaciones opuestas ya que siempre va a existir en un nivel una riostra trabajando a tracción y su pareja trabajando a compresión en función de la dirección en el momento del sismo, provocando cargas axiales cíclicas con la tendencia del comportamiento del sismo. (Duran, 2017)

Al hablar de los marcos arriostrados concéntricamente, existen dos tipos de marcos ocupados en el ámbito de las estructuras: Los marcos arriostrados concéntricos ordinario o OCBF, por las siglas en inglés correspondientes a Ordinay Concentrically Braced Frames, y los marcos arriostrados concéntricos especiales o SCBF, por sus siglas en inglés de Special Concentrically Braced Frames. A simple vista los dos tipos de marcos son iguales en cuanto a sus configuraciones, siendo su principal diferencia en el tema de las filosofías de diseño, en donde los cambios se dan en las exigencias entre uno y otro. Los marcos concéntricos especiales cuentan con requisitos más meticulosos a la hora de diseñar conexiones, vigas y columnas con respecto a los requisitos mínimos expuestos para los marcos ordinarios; teniendo como fin el garantizar que mediante el diseño siempre el primer elemento en fallar ante un evento sísmico sean las riostras y de esa forma evitar que los demás elementos estructurales sufran esfuerzos no deseados, transmitiéndose eso en un sistema en donde el daño de los elementos como vigas y columnas sea mínimo.

Siendo los diseños más estrictos los que producen que los marcos SCBF vayan a necesitar secciones para los arriostramientos en donde la sección bruta deba fluir en el estado de tracción de tal manera que ese evento se convierta en su estado límite gobernante para asegurar que se va a generar el desarrollo de ductilidad en el elemento tal como fue diseñado. Esto debe estar complementado con la restricción que se pueda producir la falla por bloque de corte en las conexiones.

Del otro lado, los marcos ordinarios OCBF al ser menos exigentes en el diseño de sus elementos sus bases de diseño de van a centrar más en la filosofía como la de los marcos normales al estar basada en la resistencia de sus elementos mediante un gran énfasis en el aumento de la rigidez de la estructura y principalmente del aporte que van a generar los arriostramientos en la misma.

1.4 Histéresis en Riostras

En orden para poder entender la filosofía de diseño y el comportamiento físico de cada riostra en función de los ciclos de cargas a los que se encuentra sometido con el fin de poder diseñar pórticos arriostrados dúctiles. Comúnmente, el comportamiento de las riostras se ve representado generalmente en función de valores de carga axial P, de la deformación axial δ y del desplazamiento transversal Δ; en donde la interacción de las variables mencionadas es lo que se conoce como diagramas histéricos, y mediante las ilustraciones 2 y 3 se puede entender el comportamiento histerético de las riostras.

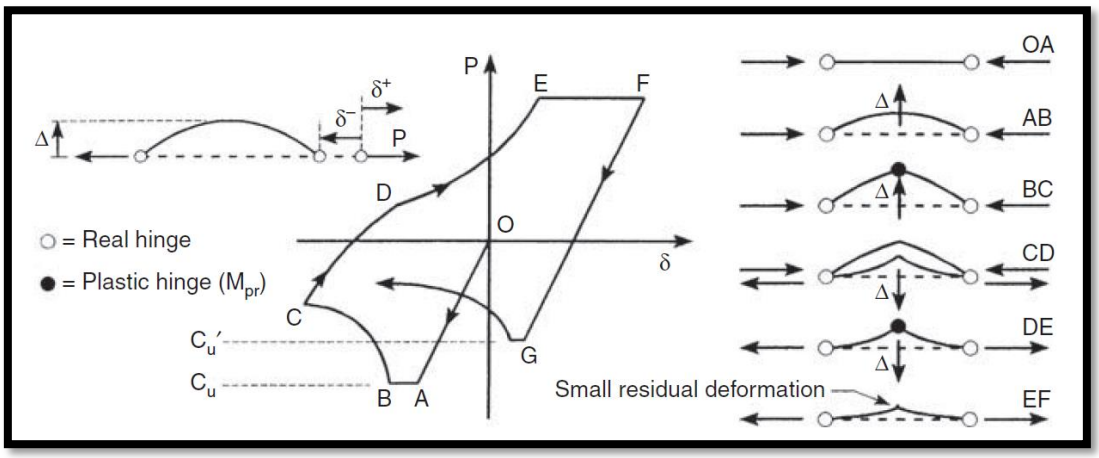


Ilustración 2: Ejemplificación 1 ciclo histerético.

Fuente: (Bruneau, Chia-Ming, & Sabelli, 2011)

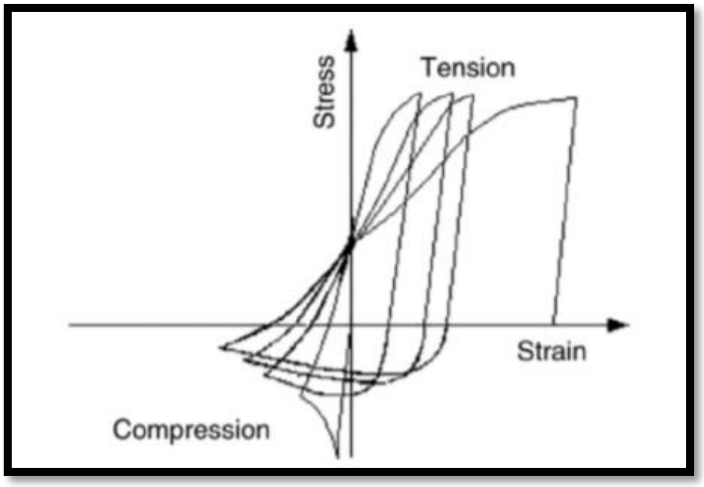


Ilustración 3: Esquema comportamiento histerético en riostras.

Fuente: (Duran, 2017)

Tomando como referencia la ilustración 2, se puede entender todos los pasos en los cuales una riostra se ve sometida al momento de sufrir 1 ciclo en el proceso de histéresis en la misma.

El punto inicial de partida o punto O es el correspondiente al estado previo a la carga del elemento. Donde, una vez empezada la carga, la riostra se comprime en el rango linealmente elástico hasta llegar al punto A donde la carga P es igual a Cu en donde se produce el pandeo del elemento. El punto A representa la etapa en la cual las riostras delgadas son capaces de soportar su carga axial aplicada cuando la riostra sufre una desviación lateral, produciendo una reducción en la deformación ante la misma carga lo cual se representa como el tramo AB en la figura 2.

El momento en que se dé el pandeo, al producirse deflexiones laterales en el elemento va a generar que se den momentos de flexión. Suponiendo un comportamiento de flexión elastoplástico bilineal, a medida que el desplazamiento transversal de la riostra aumenta aún más bajo la fuerza axial constante, se va a generar la rótula plástica en el elemento al llegar al momento plástico lo que corresponde al punto B de la gráfica.

Los desplazamientos generados por el comportamiento explicado en el párrafo anterior van a generar que estos a su vez aumente el desplazamiento transversal de la riostra, lo que va a generar rotaciones de las rótulas plásticas que es el tramo que corresponde al segmento BC, y que va a terminar en una caída de la resistencia axial del elemento porque no se puede aumentar el momento que soporta el elemento debido a la existencia ya del momento plástico. Como se observa en la ilustración, el segmento BC no tiene un comportamiento lineal por la interacción flexión-carga axial en la bisagra plástica.

Posteriormente, una vez terminada la aplicación de las cargas en el un sentido, el elemento entra en el proceso de descarga del elemento hasta el punto donde P=0. Sin embargo, al tener la riostra una deflexión axial residual δ que es irreversible, debido a rotaciones plásticas residuales.

Cuando la riostra pasa a soportar las cargas en el otro sentido, empieza el estado de tracción en el elemento y se da un comportamiento de carga – deformación lineal hasta el punto donde el producto de la deformación con la carga axial llega a generar el momento plástico en el elemento (correspondiente al punto D), produciendo al igual que en la etapa de carga la rótula plástica en el centro del elemento; con la diferencia que en esta etapa al tener la rótula que actúa inversamente en dirección del estado de carga a compresión va a reducir el desplazamiento transversal.

Ya que existe una deformación transversal irreversible, la fuerza axial en la riostra no puede exceder su resistencia a la tracción y no se pueden evitar las deflexiones transversales residuales, en donde la cedencia a la tensión se muestra como el segmento EF. Siendo este punto el final de las cargas que producen tracción en el elemento se va a descargar el elemento con una deformación residual.

Cuando se inicia el segundo ciclo en el elemento, al someterlo en un estado de compresión se va a comportar inicialmente de forma lineal, pero partiendo de la deformación remanente; lo que va a producir que la capacidad de pandeo se reduzca hasta el punto G donde se da a un valor de Cu'. La longitud de la meseta de pandeo elástico en el segmento inicial (AB) también se reduce en cada ciclo inelástico subsiguiente como resultado de la deflexión inicial residual. Más allá de estas dos diferencias, la forma de las curvas de histéresis en los ciclos inelásticos posteriores permanece básicamente sin cambios. (Bruneau, Chia-Ming, & Sabelli, 2011)

1.5 Diseño Marcos CBF

Al tratarse el Ecuador de un país en el cual el componente sísmico es el más influyente a la hora del diseño de edificaciones, se llegan a considerar a las fuerzas que va a recibir la estructura de manera sustancialmente más pequeñas en comparación a las fuerzas obtenidas si se tomara el modelo como una edificación que va a tener un comportamiento completamente elástico durante un terremoto. Esto se debe a la forma en que son diseñados los elementos estructurales, ya que estos son diseñados con el fin de tener obligadamente una respuesta dúctil en función de fuerzas más bajas y que según los fundamentos para el diseño por capacidad van a limitar a su vez a las fuerzas que se distribuyen en los elementos del sistema estructural. (Bruneau, Chia-Ming, & Sabelli, 2011)

En cuanto al diseño por capacidad, se basa en la filosofía en la cual se espera que en las estructuras se encuentren con todas las suficientes formas de disipación de energía con el fin de poder obtener las esperadas fallas dúctiles en los elementos designados para asegurar que en la estructura no se den fallas de carácter frágil que pueden generarse con en los casos en que se dé una falla frágil por corte. Es por esto que para el

diseño se debe hacer que se generen las deformaciones necesarias a flexión en las riostras a una distancia considerable como para prevenir que se den fallas por flexión en las columnas de las estructuras en sus extremos; puesto que esto puede desencadenar en el proceso de formación de mecanismos en la estructura que la volverían inestable y un riesgo para la vida de las personas. (Loges, 2017)

En función a lo mencionado, se esperaría que, durante un evento sísmico, los marcos CBF lleguen a ceder en ciertos de sus elementos diseñados para ese objetivo; y mediante el proceso se genera la disipación de la energía estimada mediante el comportamiento histérico de los elementos una vez concluido el pandeo generado por el terremoto en sus miembros. En donde los elementos de mayor incidencia van a ser las riostras existentes en los marcos de la estructura, ya que los miembros van a tener un comportamiento en función de la dirección de las fuerzas sísmicas generadas en la estructura. Puesto que algunos de las riostras en un momento van a pandearse por la compresión que se genera en los elementos; mientras que las riostras restantes van a trabajar a tensión provocando su cedencia, y que van a cambiar sus comportamientos en el momento que el sismo cambie de orientación debido a su comportamiento de carga cíclica. Esto se ve reflejado a continuación mediante la siguiente imagen que evidencia el comportamiento general de las riostras en las estructuras.

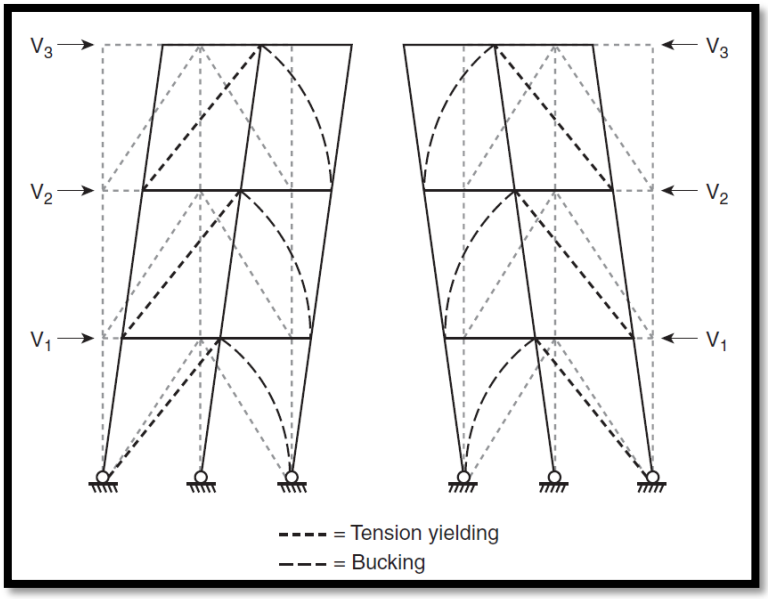


Ilustración 3: Esquema comportamiento marcos concéntricos.

Fuente: (Bruneau, Chia-Ming, & Sabelli, 2011)

En función de las condiciones mencionadas anteriormente a las que se ven sometidas las riostras, recae la importancia de diseñar de una excelente manera los elementos; ya que en función de su objetivo de mantener la seguridad ante un terremoto, los tirantes deben tener la capacidad de soportar un gran desplazamiento inelástico sin que se presente cambios considerables en la resistencia de los elementos así como evitar los posibles cambios en la rigidez de los elementos por los cambios en la rigidez que aportan a la estructura. Siendo esta la razón para un diseño especial basado en la ductilidad de esos elementos.

Para proporcionar una resistencia adecuada en los elementos ante los terremotos, los CBF deben diseñarse para alcanzar una resistencia óptima en complemento con una adecuada respuesta dúctil. Por esto, las riostras diagonales deben ser diseñados para soportar deformaciones plásticas y especialmente disipar la energía histerética producida por las cargas cíclicas de manera estable a través de las sucesivas transiciones entre los estados de pandeo en compresión y cedencia en tensión.

La filosofía de diseño va a permitir, en función de los ciclos histeréticos de las diagonales, asegurar que las columnas y vigas de las estructuras se mantengan intactas sin deformaciones plásticas, permitiendo que la estructura mantenga su capacidad ante cargas gravitacionales luego de sufrir fuertes terremotos. Hace ya algún tiempo el pensamiento sobre los principios de diseño para este tipo de estructuras fue que las riostras con baja esbeltez y con una baja relación ancho espesor van a producir un rendimiento superior en el ámbito sísmico; basado en el hecho que las riostras que en el momento de análisis se encuentran actuando a compresión van a proporcionar una disipación de energía mayor debido a la formación de la rótula plástica que se desarrolla por la falla del material en el punto de máximo momento generado. Para asegurarse que eso se diera toma importancia los bajos límites de la relación ancho espesor porque va a prevenir que se produzca una rotura frágil en la sección de la riostra por el hecho de tener un componente de la sección esbelto y que puede desarrollar un pandeo local que conduce a la generación de fatiga en el elemento. Pero si bien esta filosofía es válida en la actualidad tomando en cuenta las menores demandas de ductilidad en los miembros a compresión y confiando en las deformaciones por tensión para también disipar energía; no se toma en cuenta el hecho de la mayor diferencia existente entre el par de riostras actuando en el marco analizado con relación a la resistencia cuando una trabaja a compresión y la otra a tensión. Por lo cual en recientes ediciones se ha tomado en cuenta esas consideraciones con el fin de mantener la intención de diseño. Para compensar dicho desbalance en los marcos CBF que cuentan con una mayor ductilidad se determinó un factor de modificación de la respuesta estructural que corresponde al 75% del valor máximo asignado a los marcos, que representa la menor disipación de energía generado con la riostra en estado de compresión. Es por esto que para el diseño

de los SCBF enfatiza un rendimiento inelástico estable en la estructura con una considerable disipación de energía.

El comportamiento cíclico inelástico de un marco arriostrado sólo por tensión se caracteriza por la deformación y elongación de los arriostramientos de tensión y el pandeo de los arriostramientos de compresión a niveles de carga axial casi nulos debido a su alta delgadez. Tras una carga cíclica repetida, cada riostra acumula desplazamientos axiales residuales y el marco con riostra pierde su rigidez lateral en las proximidades del desplazamiento cero del marco. (Bruneau, Chia-Ming, & Sabelli, 2011)

1.6Ejemplo Diseño Marco CBF mediante Excel

Para poder evidenciar y optimizar el proceso de diseño marcos concéntricos con configuraciones tipo V y tipo V invertida, se procedió a la elaboración de una hoja electrónica dinámica en Excel basado en los fundamentos establecidos en la sección anterior así como tomando en cuenta la Norma Ecuatoriana de la Construcción para generar la distribución de fuerzas que van a tener que ser implementadas para el diseño tanto de riostras como también de vigas, columnas y las conexiones de las riostras con el mecanismo.

Por esto, en primer paso importante que se debía tener en cuenta para generar resultados fiables y certeros fue la toma en consideración de las variables principales establecidas en la NEC para la distribución de fuerzas sísmicas. Tal es el caso de la ubicación en la cual se planea idealizar a la estructura ya que en función de la ubicación geográfica donde se desea idealizar se van a contar con valores de peligrosidad sísmica para el ecuador según su categorización; siendo esto mostrado a continuación y siendo complementado con la gráfica puesta por la NEC donde se diferencia las zonas en función de los colores preestablecidos.

Zona Sísmica	I	II	III	IV	V	VI
Valor Factor Z	0.15	0.25	0.3	0.35	0.4	0.5
Caracterización de peligro sísmico	INTERMEDIO	ALTO	ALTO	ALTO	ALTO	MUY ALTO

Tabla 1: Valores de factor Z en función de la peligrosidad sísmica.

Fuente: (Ministerio de Desarrollo Urbano y Vivienda, 2015)

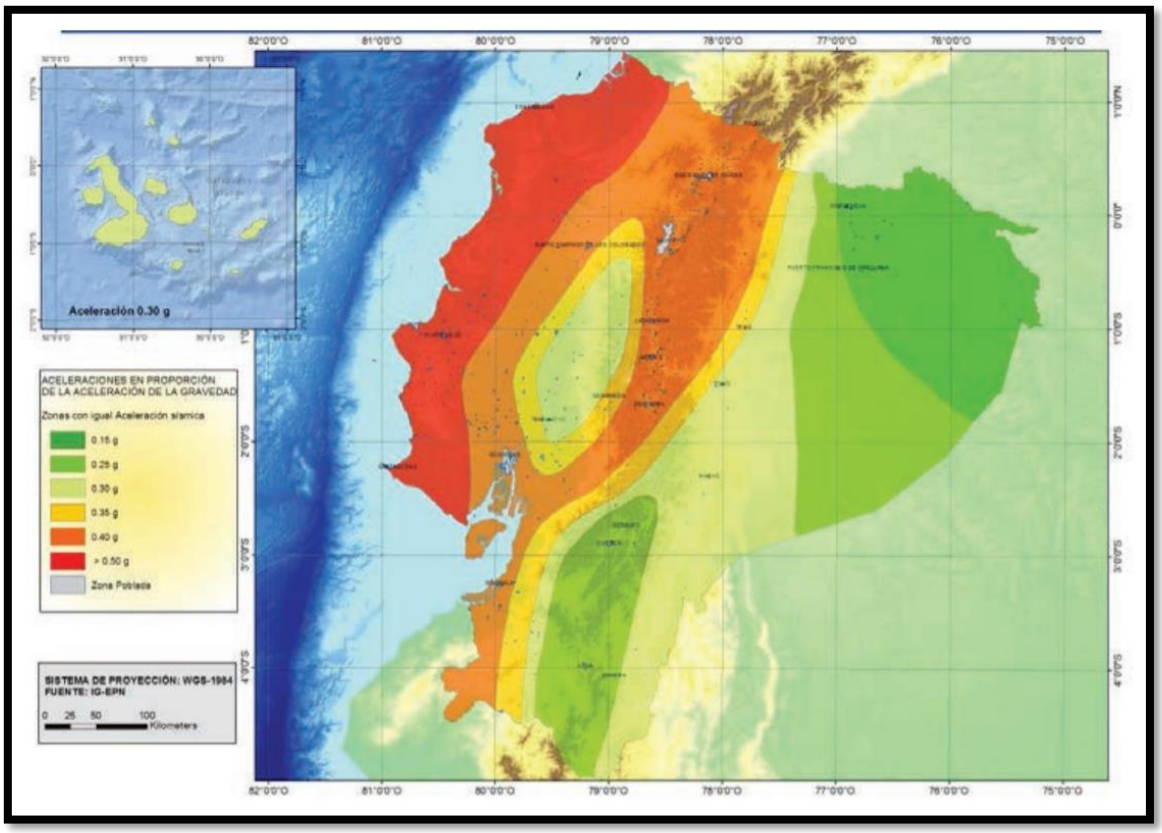


Ilustración 4: Valores de factor Z en función de la peligrosidad sísmica.

Fuente: (Ministerio de Desarrollo Urbano y Vivienda, 2015)

Tal como se muestra en la imagen anterior, los colores son los que reflejan las 6 categorías de peligro sísmico existentes en la tabla 1, por lo que el color verde más oscuro es el que representa la menor peligrosidad sísmica o zona sísmica I en donde el valor que va a recibir la estructura en función de la gravedad es del 0.15; mientras que la zona sísmica VI corresponde al color rojo en la figura la cual posee un efecto de la gravedad de 0,5 veces la misma.

De igual forma que con la distribución geográfica y su incidencia sísmica en la estructura, la NEC establece valores para poder determinar la aceleración espectral que va a sufrir la estructura, la cual se encuentra basada en coeficientes preestablecidos relacionados con las condiciones de suelo. Esto es de suma importancia porque mediante el valor a obtener en la siguiente imagen se va a determinar el cortante basal que va a soportar la estructura en su base y posteriormente ser distribuida por la estructura.

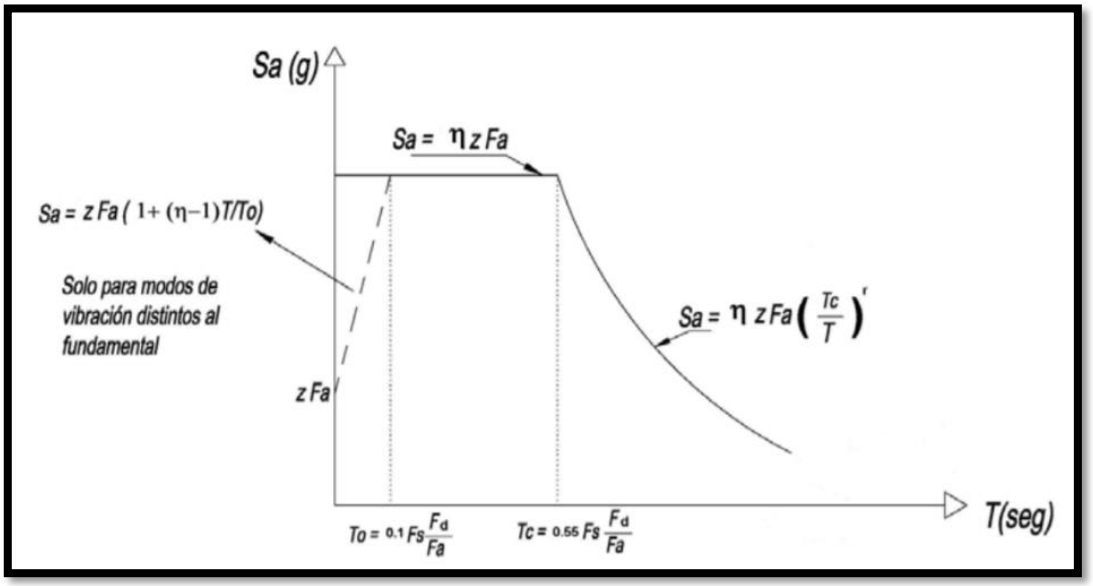


Ilustración 5: Espectro de diseño en aceleraciones.

Fuente: (Ministerio de Desarrollo Urbano y Vivienda, 2015)

Tomando como referencia la figura 5, los componentes del espectro dependen de factores que se encuentran a su vez normados dependiendo de las condiciones de suelo en las que se encuentra la estructura.

COEFICIENTE DE AMPLIFICACIÓN DEL SUELO FA						
TIPO DE SUELO	ZONA SÍSMICA					
	I	II	III	IV	V	VI
A	0.9	0.9	0.9	0.9	0.9	0.9

B	1	1	1	1	1	1
C	1.4	1.3	1.25	1.23	1.2	1.18
D	1.6	1.4	1.3	1.25	1.2	1.12
E	1.8	1.4	1.25	1.1	1	0.85

Tabla 2: Coeficiente de Amplificación del Suelo (Fa)

Fuente: (Ministerio de Desarrollo Urbano y Vivienda, 2015)

COEFICIENTE DE AMPLIFICACIÓN DE ORDENADAS DEL ESPECTRO ELASTICO FD						
TIPO DE SUELO	ZONA SÍSMICA					
	I	II	III	IV	V	VI
A	0.9	0.9	0.9	0.9	0.9	0.9
B	1	1	1	1	1	1
C	1.36	1.28	1.19	1.15	1.11	1.06
D	1.62	1.45	1.36	1.28	1.19	1.11
E	2.1	1.75	1.7	1.65	1.6	1.5

Tabla 3: Coeficiente de Amplificación de Ordenadas del Espectro Elástico (Fd)

Fuente: (Ministerio de Desarrollo Urbano y Vivienda, 2015)

COEFICIENTE DE COMPRTAMIENTO NO LINEAL FS						
TIPO DE SUELO	ZONA SÍSMICA					
	I	II	III	IV	V	VI
A	0.75	0.75	0.75	0.75	0.75	0.75
B	0.75	0.75	0.75	0.75	0.75	0.75
C	0.85	0.94	1.02	1.06	1.11	1.23
D	1.02	1.06	1.11	1.19	1.28	1.4
E	1.5	1.6	1.7	1.8	1.9	2

Tabla 4: Coeficiente de Comportamiento No Lineal (Fs)

Fuente: (Ministerio de Desarrollo Urbano y Vivienda, 2015)

Tal como se puede evidenciar en las tres tablas anteriores, los valores de los coeficientes de Fd, Fs y Fa están determinados en función de la zona sísmica explicada anteriormente, así como en función de la clasificación del tipo de suelo existente en la NEC. En donde, el tipo de suelo se encuentra clasificado por ciertas características presentes en los mismos; siendo el suelo tipo A el correspondiente a un perfil de roca

competente que correspondería a la mejor categoría de suelo que se podría encontrar en el Ecuador. Mientras que el suelo tipo B corresponde a un de roca de rigidez media; el suelo C es la categoría de perfiles de suelo muy densos o roca considerada blanda, el suelo D corresponde a un suelo rígido de menor calidad que el C. Finalmente, los suelos E y F son los de peor calidad en la clasificación en donde se encuentra más componentes como arcillas dentro de las mismas.

Por otro lado, la razón espectral corresponde a la relación entre la aceleración espectral para un periodo establecido de 1 segundo con respecto al PGA (Peak Ground Acceleration) correspondiente al mismo periodo. En el cual para facilidad de diseño se obtuvieron valores promedios para las generalizar los valores para las regiones del Ecuador.

RAZON ACELERACIÓN ESPECTRAL η	
COSTA	1.8
SIERRA	2.48
ORIENTE	2.6
INSULAR	2.48

Tabla 5: Razón Aceleración Espectral en Función de la Región

Fuente: (Ministerio de Desarrollo Urbano y Vivienda, 2015)

Complementario a las constantes establecidas, es necesario determinar otros parámetros necesarios para determinar las fuerzas sísmicas en cada piso. Entre las variables que son necesarias determinar involucran el número de pisos de la estructura, la altura de entre piso, el peso esperado de toda la estructura y al ser estructuras de acero es necesario estableces dos constantes para poder estimar el periodo de la estructura.

Habiendo determinado los valores establecidos, el posible determinar el cortante basal de la estructura, basándose en la ecuación presentada en la NEC a continuación:

$$V = \frac{I\,Sa(Ta)}{R\,\emptyset_P\emptyset_E}W$$

Por lo cual, tomando en cuenta el ejemplo que se realizó en Excel se obtuvieron los siguientes valores en función de los datos ingresados:

PROYECTO DE TITULACIÓN

Design of Special Concentrically Braced Frame

PEDRO JOSÉ HIDALGO ORTEGA

USFQ

DATOS A INGRESAR

DATOS DEL PROYECTO	
UBICACION	
PROVINCIA	Santa Elena
REGION	COSTA

Carga Muerta Sobreimpuesta	
Item	Peso (ton/m2)
Paredes	0.18
Recubrimiento de Paredes	0.015
Recubrimiento de pisos	0.06
Instalaciones sanitarias, electric	0.06
Misceláneos	0.01
Carga Viva	
NEC-15	0.2

Datos Estructura	
# de Pisos	6
Altura Entrepisos	3.65
Peso Total de la Estructura (SAP) [tonf]	6453.96
Periodo de la Estructura	
c_t	0.02
h_n [m]	21.9
α	0.75
T [s]	0.20

Datos Suelo	
Zona Sísmica	V
Tipo de Suelo	B

Ilustración 6: Datos modelo ejemplo diseño.

Fuente: Propia.

CORTANTE BASAL Y COEFICIENTE SISMICO

Zona sismica	V	η	1.8
Faxtor Z	0.4	r	1
Peligro Sismico	ALTO	Sa	0.72
Fa	1	I (importancia)	1.00
Fd	1	φp	1.00
Fs	0.75	φe	1.00
To	0.075	R	8.00
Tc	0.413	V [tons]	580.86
		Coeficiente Vbase	0.09

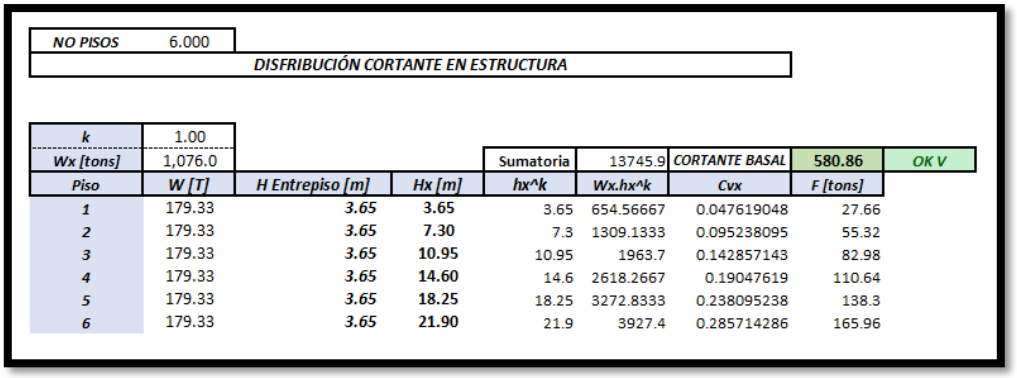
$$V = \frac{IS_a(T_a)}{R\emptyset_P\emptyset_E}W$$

Ilustración 7: Determinación Cortante Basal Ejemplo Base.

Fuente: Propia.

Posteriormente a la determinación de cortante basal se lo debe distribuir a lo largo de cada piso para saber las fuerzas que van a actuar en los elementos de cada

piso para lo cual se creó un macro en la hoja electrónica que permite generar la distribución de fuerzas automática en función de los valores generales ingresados; pero que el código se encuentra en la sección de anexos en el apartado A.



NO PISOS	6.000							
DISFRIBUCIÓN CORTANTE EN ESTRUCTURA								
k	1.00							
Wx [tons]	1,076.0			Sumatoria	13745.9	CORTANTE BASAL	580.86	OK V
Piso	W [T]	H Entrepiso [m]	Hx [m]	hx^k	Wx.hx^k	Cvx	F [tons]	
1	179.33	3.65	3.65	3.65	654.56667	0.047619048	27.66	
2	179.33	3.65	7.30	7.3	1309.1333	0.095238095	55.32	
3	179.33	3.65	10.95	10.95	1963.7	0.142857143	82.98	
4	179.33	3.65	14.60	14.6	2618.2667	0.19047619	110.64	
5	179.33	3.65	18.25	18.25	3272.8333	0.238095238	138.3	
6	179.33	3.65	21.90	21.9	3927.4	0.285714286	165.96	

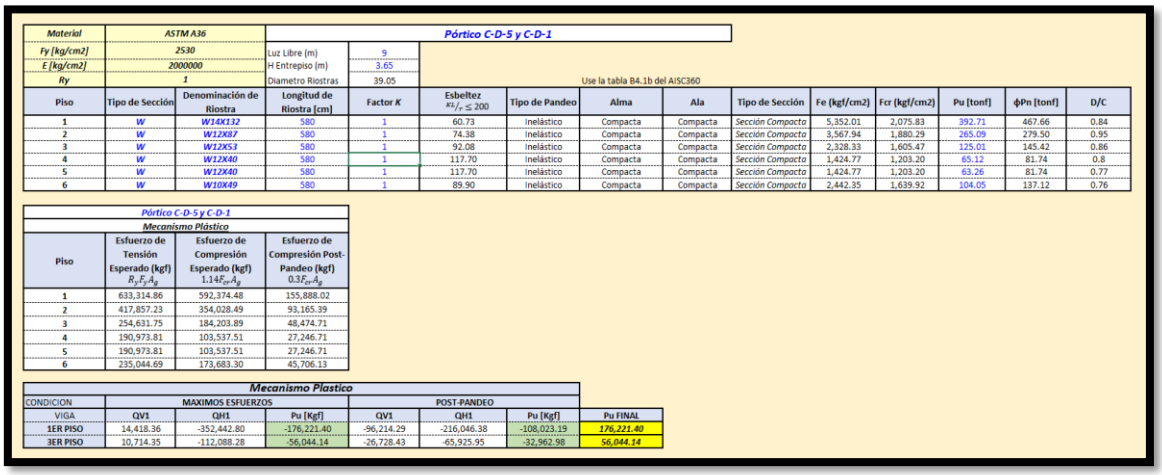
Ilustración 8: Ejemplo distribución de fuerzas para los pisos analizados

Fuente: Propia

Mediante la ilustración 8 se puede ver cómo se distribuyen las fuerzas horizontales equivalentes a la carga sísmica en cada uno de los pisos, manteniendo el comportamiento ascendente del valor de la fuerza conforme se va analizando los pisos superiores. Sin embargo, para poder pasar al análisis completo para el diseño de los elementos de la estructura es necesario obtener también las cargas axiales que van a sufrir los elementos pertenecientes a las columnas y riostras principalmente; lo cual nos lleva a una de las limitaciones en el proceso de automatización para el diseño del sistema estructural. Esto se debe a que los valores de las fuerzas axiales que van a sufrir los elementos y en concreto de las riostras solo se puede obtener a partir de algún software de análisis estructural alternativo que nos despliegue todos los esfuerzos que se encuentran sometidos los elementos. Por lo cual esta parte del análisis se volvería en un tema manual a realizar por cada persona en donde se debe ingresar los valores de las cargas axiales principalmente. En consecuencia, la primera parte del propósito del

programa se encontraría terminado ya que con esos valores se debe realizar el análisis de la estructura aparte para poder continuar con la segunda función de la hoja elaborada.

Para la segunda parte contenida en el programa de Excel, su propósito es poder interactuar con las secciones existentes en los catálogos del AISC para poder obtener la sección óptima para las riostras de cada piso con el fin de soportar la demanda proveniente del análisis y que sea lo más económica posible. En el proceso de lograr el objetivo, se debe analizar a la estructura en cada uno de sus sentidos principales, tal como se evidencia en la ilustración 9.



Material	ASTM A36	Pórtico C-D-5 y C-D-1	
Fy [kg/cm2]	2530	Luz Libre (m)	9
E [kg/cm2]	2000000	H Entrepiso (m)	3.65
Ry	1	Diametro Riostras	39.05

Use la tabla B4.1b del AISC360

Piso	Tipo de Sección	Denominación de Riostra	Longitud de Riostra [cm]	Factor K	Esbeltez $\frac{Kl_c}{r} \leq 200$	Tipo de Pandeo	Alma	Ala	Tipo de Sección	Fe (kgf/cm2)	Fcr (kgf/cm2)	Pu [tonf]	ϕPn [tonf]	D/C
1	W	W14X132	580	1	60.73	Inelástico	Compacta	Compacta	Sección Compacta	5,352.01	2,075.83	392.71	467.66	0.84
2	W	W12X87	580	1	74.38	Inelástico	Compacta	Compacta	Sección Compacta	3,567.94	1,880.29	265.09	279.50	0.95
3	W	W12X53	580	1	92.08	Inelástico	Compacta	Compacta	Sección Compacta	2,328.33	1,605.47	125.01	145.42	0.86
4	W	W12X40	580	1	117.70	Inelástico	Compacta	Compacta	Sección Compacta	1,424.77	1,203.20	65.12	81.74	0.8
5	W	W12X40	580	1	117.70	Inelástico	Compacta	Compacta	Sección Compacta	1,424.77	1,203.20	63.26	81.74	0.77
6	W	W10X49	580	1	89.90	Inelástico	Compacta	Compacta	Sección Compacta	2,442.35	1,639.92	104.05	137.12	0.76

Pórtico C-D-5 y C-D-1

Mecanismo Plástico

Piso	Esfuerzo de Tensión Esperado (kgf) $R_y F_y A_g$	Esfuerzo de Compresión Esperado (kgf) $1.14 F_{cr} A_g$	Esfuerzo de Compresión Post-Pandeo (kgf) $0.3 F_{cr} A_g$
1	633,314.86	592,374.48	155,888.02
2	417,857.23	354,028.49	93,165.39
3	254,631.75	184,203.89	48,474.71
4	190,973.81	103,537.51	27,246.71
5	190,973.81	103,537.51	27,246.71
6	235,044.69	173,683.30	45,706.13

Mecanismo Plastico

CONDICION	MAXIMOS ESFUERZOS			POST-PANDEO			
VIGA	QV1	QH1	Pu [Kgf]	QV1	QH1	Pu [Kgf]	Pu FINAL
1ER PISO	14,418.36	-352,442.80	-176,221.40	-96,214.29	-216,046.38	-108,023.19	176,221.40
3ER PISO	10,714.35	-112,088.28	-56,044.14	-26,728.43	-65,925.95	-32,962.98	56,044.14

Ilustración 9: Hoja Electrónica Diseño Riostras Ejemplo Base

Fuente: Propia.

Tal como se puede evidenciar en la imagen 9, se observa todos los pasos que se deben seguir para realizar el diseño de los tirantes. Donde primero se debe seleccionar el tipo de sección de las cuales van a ser hechas las riostras, generalmente van a ser secciones tipo W, y se debe seleccionar una denominación de la sección en función de los catálogos incluidos en el archivo Excel con todas sus propiedades. Habiendo elegido una sección inicial para el proceso de iteración, es importante determinar la longitud de las riostras en cuestión que van a estar basadas en la longitud de entrepiso como de la dimensión de luz libre entre columnas de cada piso. Finalmente, en la etapa de

especificación de las condiciones iniciales de cada riostra se debe definir las condiciones de borde que va a tener la riostra en sus extremos, que para este caso va a ser condiciones de borde solo conectadas tipo articulaciones para generar que solo existan fuerzas axiales en los elementos.

Posterior a la definición, se debe calcular la esbeltez correspondiente a cada riostra elegida, para poder determinar el tipo de pandeo al cual se encontraría sometido. Adicionalmente se debe chequear los componentes de cada sección y ver si las mismas cuentan con el alma y los patines compactos para saber si la sección puede ser clasificada como una sección compacta. Con el tipo de sección determinada se debe calcular la capacidad de la sección, la cual se basa en los valores de Fcr, para finalmente comparar con los valores demandados para cada riostra y obtener su relación demanda – capacidad.

Por otro lado, el diseño de las vigas y columnas se ven presentados en la ilustración 10 en donde se realizan el diseño de los elementos del primer piso. Para realizar estos diseños al igual que para la riostra es necesario definir una sección inicial de análisis en donde se le debe chequear la estabilidad local del elemento. Se debe revisar que los componentes de la sección sean compactos, posteriormente se debe obtener la esbeltez del elemento para en función de eso poder determinar su capacidad a compresión la cual debe ser comparada con las cargas axiales provenientes del análisis del mecanismo plástico de la viga tomando en cuenta la configuración de la riostra diseñada previamente; lo cual nos da como resultado la demanda – capacidad a compresión del elemento. Esto debe ser complementado con el chequeo del elemento en su capacidad de flexión; ya que al ser estructuras de acero son propensas a sufrir el fenómeno de pandeo lateral torsional (PLT).

Para el caso de vigas este chequeo se debe hacer en función de los valores obtenidos del momento último que debe soportar la viga del análisis estructural. El cual debe ser comparado con el momento nominal calculado en función del estado de pandeo lateral torsional que tenga el elemento y de esta forma poder obtener la demanda – capacidad. Mientras que para las columnas se debe implementar el diagrama de interacción entre carga axial y momento; donde se determina el momento nominal de la misma forma que la viga en función del pandeo existente en el elemento y se debe revisar que en la interacción del momento con la carga axial se obtenga un valor menor a 1 basándose en la tabla B4.1b del AISC 360.

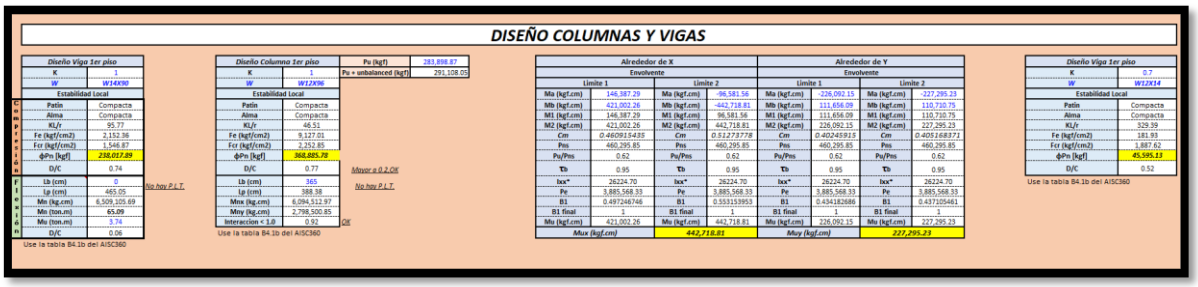


Ilustración 10: Diseño vigas y columnas ejemplo diseño.

Fuente: Propia

Finalmente, para el proceso de diseño de las conexiones es primordial establecer el tipo de material con el cual se va a realizar la conexión, que son de tipo Gusset Plate, así como el material de la soldadura y del electrodo a utilizar. En el proceso de diseño de la placa de conexión, se va a tener en cuenta el valor del esfuerzo a tensión esperado obtenido en el análisis de la riostra, en donde se va a establecer un espesor de la placa, así como también el tamaño de la soldadura para poder determinar la longitud en centímetros que debe tener la placa para poder realizar la conexión esperada siendo esto evidenciado en el proceso realizado en la hoja electrónica en la imagen 11.

DISEÑO CONEXIONES

Soldadura Knife Plate to Gusset

Material Gusset plate	*ASTM A36*		
Fy (kg/cm2)	*2530*		
Ry	*1*		
Material	*ASTM A572*	Material Electrodo	*ASTM A572*
Fy (kg/cm2)	*3515*	Fy (kg/cm2)	*3515*
Electrodo	*E60XX*	Electrodo	*E60XX*
FExx (kgf/cm2)	*3520*	FExx (kgf/cm2)	*3520*

Nivel	Tipo de Sección	Denominación de Riostra	$R_{u1}\ (kgf)$	t_{kp} *(cm)*	S Tamaño de Soldadura (cm)	l_w *(cm)*	Espesor del Gusset Plate (tg) [cm]	S Tamaño de Soldadura (cm)	l_w *(cm)* l_w *(cm)*
1	W	W14X132	633314.8624	4	2	59	5	3.138	37.54489658
2	W	W12X87	417857.2288	4	2	39	5	3.138	24.77189052
3	W	W12X53	254631.7488	3	2	24	4	2.51	18.87222053
4	W	W12X40	190973.8116	3	2	18	4	2.51	14.1541654
5	W	W12X40	190973.8116	3	2	18	4	2.51	14.1541654
6	W	W10X49	235044.6912	3	2	22	4	2.51	17.42051126

Ilustración 11: Diseño conexiones de riostras con marco.

Fuente: Propia.

Teniendo diseñado las riostras, los elementos de los marcos y las conexiones se puede dar por terminado el proceso de diseño en donde todos los elementos en cuestión cumplen con la demanda esperada en la estructura.

Capítulo 2: Análisis No Lineal en Estructuras

2.1Definición

Cuando nos referimos al análisis estructural no lineal, nos referimos a uno de los procesos más importantes a la hora de evaluación y diseño sísmico en el contexto del diseño por desempeño. Para el proceso general de poder realizar el análisis no lineal de las estructuras, se deben tener en consideración ciertas pautas. El primer punto a tener en consideración es que se debe tener definido de manera clara cuál va a ser el objetivo de realizar el análisis no lineal ya que mediante esto se debe tenar claro cuáles son los resultados que se desean obtener del análisis y comportamiento del edificio.

Otro punto a considerar es entender cómo se espera que sea el comportamiento de la estructura que se va a estar analizando, así como los modos de falla que se esperaría obtener. Lo cual tiene como objetivo el poder tomar en cuenta todas las posibles repercusiones de dichos comportamientos en la respuesta estructural obtenida en el análisis no lineal y su incidencia en los datos a analizar en la evaluación de su desempeño en la etapa de post – procesamiento de datos.

Complementariamente con la necesidad de definir los parámetros de demanda, así como establecer los criterios de aceptación específicos para la estructura a analizar. Siendo los parámetros de demanda generalmente las derivas de pisos máximas, aceleraciones máximas de piso o las deformaciones inelásticas en elementos.

En función de la precisión necesaria en los resultados obtenidos en el análisis no lineal de la estructura, se debe determinar los parámetros mínimos necesarios en el análisis en función del nivel de detalle. Lo cual va desde generar los modelos más simples y prácticos como modelos de tipo bisagra plástica hasta tener un nivel de detalle supremamente mayor como en la realización del modelado mediante la aplicación de

elementos finitos continuos. Estas distinciones dependen de las características de la estructura y del uso de conceptos de diseño de capacidad para controlar las ubicaciones de inflexibilidad. La selección del tipo de modelo y los parámetros de modelado para componentes controlados por deformación también puede depender de la extensión de las deformaciones inelásticas esperadas. Las incertidumbres en los parámetros del modelo y la respuesta pueden explicarse mediante el uso de análisis de sensibilidad. (National Institute of Standards and Technology, 2017)

Finalmente, habiendo predeterminado las cuestiones anteriores se procede a la realización de los análisis pertinentes. Siendo en función del tipo de diseño que se está basando la estructura. Los análisis no lineales se realizan bajo cargas de gravedad apropiadas, otras cargas no sísmicas y efectos de carga sísmica; y sus efectos se van a ver modelados mediante los tipos de análisis estáticos y dinámicos, los cuales van a ser explicados más a detalle a continuación en las siguientes secciones.

2.2Análisis No Lineal Estático Pushover

El primer tipo de análisis que se debe realizar al realizar un análisis no lineal es el de tipo elástico el cual es denominado Pushover; el cual tiene como su objetivo principal el aproximar el comportamiento dinámico de la estructura como respuesta de los movimientos del suelo producidos por un terremoto mediante la aplicación de una carga lateral estática equivalente.

Es un componente del análisis de estructuras que en el pasado no era una opción muy viable de realización debido a las demandas computacionales que este requiere, así como las limitaciones en cuanto a softwares existentes en la época que todavía no se desarrollaban en una gran manera los métodos de análisis relacionados con el comportamiento del suelo. Pero, en la actualidad, esa situación ha cambiado ya que el

análisis dinámico no lineal ahora puede aplicarse de forma más rutinaria en los procesos ingenieriles; siendo principalmente por los avances computacionales, así como la caracterización de los riesgos sísmicos. Por lo tanto, la mayor disponibilidad y practicidad del análisis dinámico no lineal, combinado con la inherente limitación del análisis estático no lineal, han reducido drásticamente la dependencia de análisis estático no lineal para el diseño. (National Institute of Standards and Technology, 2017)

Sin embargo, a pesar de la presencia del análisis dinámico no lineal en las estructuras, el análisis estático no lineal sigue teniendo un papel fundamental al ser un complemento del otro tipo de análisis. En particular, debido a la mayor eficacia del análisis estático para poder verificar los modelos determinados para el análisis antes de ejecutar el análisis dinámico; así como para comprender su comportamiento no lineal y de esta forma poder afinar detalles en cuanto a los diseños sísmicos de la estructura. Complementariamente, si se está analizando estructuras de poca altura, el análisis estático no lineal puede ser suficientemente preciso para poder diseñar mediante la filosofía sísmica; esto por tener su respuesta estructural dominada por modo fundamental de la estructura.

El procedimiento para poder realizar un análisis estático no lineal con orientación sísmica se encuentra definido y descrito mediante el documento del ASCE/ SEI 41-13. Siendo contemplado primero la creación y calibración del modelo a analizar. Posteriormente es mandatorio el definir los patrones de cargas gravitacionales estáticas que van a actuar en la estructural. Al igual que en cualquier de proceso de diseño sísmico se debe determinar las cargas laterales equivalentes para poder simular los efectos de los sismos de diseño. Adicionalmente se debe establecer cual va a ser el desplazamiento esperado como objetivo en función de la intensidad de terremoto a analizar. Con lo cual se realiza el proceso de pushover que consiste en ir incrementando

gradualmente la carga lateral equivalente hasta hacer que se produzca el desplazamiento objetivo en la estructura. Habiendo alcanzado el desplazamiento objetivo, se debe a su vez evaluar las derivas de piso generadas por las fuerzas aplicadas; así como otros parámetros relacionados a las respuestas de los elementos y las fuerzas a las cuales se encuentran sometidos. Todos los pasos mencionados pueden ser repetidos tantas veces se deseen para analizar los modelos, así como para realizar el análisis de distintos sismos.

Es importante a su vez hablar sobre los requisitos de modelado de componentes en el análisis estático no lineal, siendo los requisitos más simples que para el análisis dinámico no lineal ya que el análisis estático no necesita capturar la respuesta histerética cíclica de los elementos. Sin embargo, el comportamiento de fuerza-deformación debe calibrarse para incorporar la degradación cíclica cuando el fin principal del estático es verificar el diseño sísmico. Esto implica calibrar los modelos de componentes de rótulas concentrados para la curva de esqueleto cíclico, en lugar de la curva de esqueleto monótona. En particular, gran parte de modelos basados en los elementos finitos continuos y de tipo de fibra se formulan para simular directamente los efectos de degradación cíclica. (National Institute of Standards and Technology, 2017)

Con respecto a la carga estática equivalente, se trata de un tema de mayor complejidad en el análisis estático no lineal por las limitaciones que esto representa. En teoría, la carga estática equivalente distribuida en cada piso de la estructura debe simular a la final una estructura desplazada que sea lo más semejante posible al comportamiento en el desplazamiento real producido por la carga dinámica cíclica. Pero que, en la vida real, esto es relativamente difícil de que se produzca en condiciones que no sean las favorables, siendo lo óptimo en estructuras de baja altura o que no cuenten con irregularidades en su apariencia, porque en estas estructuras su deformada elástica

es similar a su modo de vibración fundamental elástico. Razón por la cual en ocasiones se emplean patrones de carga alternativos, como un patrón de carga uniforme en la altura del edificio. Sin embargo, debido a estas dificultades de simulación es que se prefiere el análisis dinámico no lineal al análisis no lineal estático.

Históricamente, existen dos métodos principales para determinar el desplazamiento de la estructura en función de la intensidad del movimiento del suelo y el período fundamental de vibración. Siendo el primero el denominado método de coeficiente, que es el método principal implementado en el ASCE / SEI 41; mientras que el segundo método es el llamado método de espectro de capacidad empleado en el ATC-40 y otros documentos. La diferencia principal entre los dos métodos es que el método del coeficiente tiene una mayor facilidad de aplicación mientras que el método del espectro de capacidad tiene la ventaja de poder proporcionar una interpretación gráfica de la respuesta. Indiferentemente del método que se aplique es primordial referenciar que para ambos métodos es necesario establecer una serie de suposiciones y parámetros simplificadores, los cuales son conceptos empíricos obtenidos con el paso del tiempo y son basados con datos obtenidos del análisis dinámico no lineal. Entre los dos, se recomienda el método de coeficientes de ASCE / SEI 41, ya que incorpora estudios recientes para mejorar su calibración para efectos de degradación cíclica. (National Institute of Standards and Technology, 2017)

Como se señaló anteriormente, la interpretación de los resultados del modelo de análisis estático no lineal depende de su aplicación prevista. En los casos en que se considere que el análisis proporciona una base confiable para verificar el desempeño estructural, los resultados del análisis pueden usarse para evaluar la aceptación apropiada o los criterios de estado límite. Para lo cual, los criterios de aceptación de componentes mediante el análisis estático no lineal en ASCE / SEI 41-13 se basan en

función de objetivos de rendimiento específicos como los IO, LS o CP en elementos que se encuentran controlados por deformación y fuerza. Si bien, los procedimientos para verificar los criterios de aceptación de los componentes pueden ser relativamente sencillos, la interpretación de las respuestas obtenidas no necesariamente es igual de simple; más bien tienden a complicarse. Aparte de las limitaciones inherentes del análisis estático en sí, estas ambigüedades en la relación de la respuesta del componente local con la respuesta general son otra limitación significativa que afecta tanto a los métodos de análisis estáticos como dinámicos no lineales. (National Institute of Standards and Technology, 2017)

Si bien no se puede decir que oficialmente existan criterios base de aceptación formales con relación al proceso del análisis estático no lineal para poder comprender su comportamiento inelástico, es útil tener en cuenta ciertas sugerencias a revisar en los resultados del análisis estático no lineal. Algo a tener en consideración es que para poder revisar si se está produciendo cualquier concentración de respuesta inelástica en la estructura, incluido la torsión del edificio, se deberían graficar las relaciones de las derivas de piso de la estructura regidas por varios intentos bajo los diversos desplazamientos objetivos. Adicionalmente, es recomendable comparar la deriva producida en la cubierta con relación con la respuesta al cortante basal, para poder verificar si tanto el cortante basal producido por una deformación significativa o si las cargas máximas producidas se encuentran dentro de los límites esperados.

Otra sugerencia tiene que ver con el graficar la respuesta fuerza-deformación inelástica con respecto a los elementos a analizar que hayan sido diseñados para ser controlados mediante deformación con el fin de poder comprobar que la respuesta que se genera en los elementos sea la esperada. Así mismo, es considerable verificar las

demandas de los elementos controlados por fuerza porque de esta forma es posible establecer la relación con las resistencias esperadas de los miembros.

Con respecto al análisis pushover es recomendable realizar el mismo tomando en cuenta las condiciones en donde se presentan efectos geométricos no lineales asociados a los efectos P-Delta, así como el análisis donde se desprecian estos efectos; con el fin de poder corroborar efectivamente la magnitud en que estos efectos inciden en la respuesta en el rango elástico e inelástico. Finalmente, se debería llevar a la estructura a analizar a generar desplazamientos mayores al objetivo establecido para de esta forma poder estar seguros que la estructura continúa comportándose de manera estable y es menos sensible a las variabilidades en respuesta a las incertidumbres en los movimientos del suelo y otros efectos.

No hay como dejar de lado las principales limitaciones del análisis estático no lineal, ya que estas limitaciones conducen a que se produzca en la estructura una incapacidad inherente para capturar algunas respuestas dinámicas, entre las cuales es importante mencionar los problemas para representar el comportamiento dinámico multimodo, que tiene suma importancia en edificaciones de alturas relativamente altas o en estructuras con irregularidades geométricas o estructurales. Aparte se encuentra dificultades en representar el comportamiento cíclico y degradación de los materiales. De igual forma, se debe entender que cuenta con limitaciones para representar los efectos de la relación de dependencia de los elementos, lo cual es particularmente importantes para los sistemas que emplean amortiguadores viscosos. Así mismo, el comportamiento de sistemas oscilantes o aislados sísmicamente no tienen la representación adecuada. Características únicas de los movimientos del suelo de los terremotos, incluida la forma espectral, el pulso cercano a la falla y los efectos de

duración se encuentras también dentro de las limitaciones encontradas en este modelo de análisis.

Como punto final, los métodos de análisis estático no lineal no son adecuados para evaluar las implicaciones de la respuesta de degradación de la resistencia en el sistema estructural general, lo que en última instancia puede conducir a la concentración y acumulación de desviaciones, lo que conduce a la inestabilidad estructural. (National Institute of Standards and Technology, 2017)

2.3Análisis Dinámico Tiempo-Historia

EL segundo tipo de análisis, el dinámico, es el que viene representado mediante el análisis del historial de respuestas no lineales es un análisis dinámico o llamado análisis tiempo – historia. El cual busca que el modelo estructural se someta a uno o más registros sísmicos previamente seleccionados para poder ser escalados en función de los objetivos propuestos en el análisis. Este método es un gran complemento para el análisis estático no lineal, ya que el análisis tiempo – historia supera la mayoría de las limitaciones del análisis estático no lineal; debido a su capacidad para representar con una gran precisión el comportamiento multimodo, así como los efectos dependientes de la velocidad, el comportamiento cíclico dependiente de la trayectoria, y un modelado realista de las características del movimiento del suelo, como la forma espectral, los efectos de pulso de larga duración y casi de falla. (National Institute of Standards and Technology, 2017)

A diferencia del análisis estático no lineal, el análisis tiempo – historia representa una mejor respuesta temporal de la estructura a los movimientos sísmicos producidos en el suelo. Para obtener resultados significativos, es importante que sismos implementados en los estudios reflejen con la mayor precisión posible el peligro

sísmico existente en cada sitio. Por lo cual es primordial tomar en cuenta las siguientes pautas al respecto. Se debe tomar una decisión sobre el método por el cual se va a realizar la evaluación, siendo el método más usado el método de evaluación denominado evaluación basada en intensidad que se basa en el análisis de la respuesta de la estructura para un nivel específico de sismo. Existen igual otros dos métodos menos comunes que se evaluaciones basadas en escenarios y evaluaciones basadas en tiempo.

Adicional, se debe decidir acerca de los espectros que se busca implementar y que mediante el ASCE / SEI 7-16 se establece la posibilidad de elegir entre dos tipos de espectros objetivo para una evaluación basada en la intensidad; el primero un espectro básico uniforme de peligro y los espectros específicos del sitio que tienden a ser al menos dos. Cabe recalcar el hecho que el uso de espectros múltiples va a producir que el esfuerzo de análisis se vea considerablemente aumentado generalmente en un factor superior a 2; pero que a su vez va a resultar en una reducción en conservaciones innecesarias en parámetros de las demandas sísmicas generadas por los sismos ya que cada espectro analizado va a estar envuelto por el espectro objetivo.

Otro aspecto en donde se deben tomar ciertas decisiones tiene que ver con aspectos relacionados con el suelo de análisis. Siendo en primer ligar el número de movimientos del suelo producidos por el sismo, en donde el ASCE / SEI 7-16 establece un mínimo de 11 movimientos para poder establecer con cierta veracidad la respuesta media de la estructura sometida a los movimientos del suelo en términos del máximo sismo considerado en función del riesgo objetivo o MCEr por sus siglas en inglés para cada escenario establecido. Si bien este es el valor recomendado para gran parte de las estructuras, este valor puede ser menor en casos donde las intensidades en el suelo son más bajas y por ende se va a dar una menor no linealidad en la estructura; mientras que

si la precisión requerida para el análisis es más requerida para comprender a mayor detalle la variabilidad de las respuestas, se va a necesitar realizar al menos 30 movimientos que deben ser complementados con modificaciones en la escala a realizar para poder representar de una manera más explícita la variabilidad del movimiento del suelo. En la mayoría de los casos, se utilizan únicamente las dos componentes horizontales del movimiento del suelo; sin embargo, si el edificio es sensible a los movimientos verticales, se debe incluir un componente vertical del movimiento ose debe realizar un estudio para verificar que el efecto no es significativo. Los movimientos del suelo se deben escalar de manera que los espectros coincidan con el espectro objetivo durante el rango de período importante para la respuesta del edificio. (National Institute of Standards and Technology, 2017)

La coincidencia espectral es un enfoque viable para desarrollar movimientos del suelo para análisis no lineales cuando el objetivo es predecir la respuesta media del edificio. Debido a que la coincidencia espectral puede causar una modificación significativa de la frecuencia de la señal de registro y el contenido de energía, se debe tener cuidado para garantizar que se conserven las características de movimiento importantes después de que se haya completado el proceso de coincidencia. Una posible desventaja de la coincidencia espectral ajustada es que puede reducir o eliminar la variabilidad en el conjunto de movimientos del suelo. Esto puede conducir a resultados inesperados o poco conservadores si el análisis tiene la intención de considerar la respuesta estructural del registro individual además de la respuesta media. (National Institute of Standards and Technology, 2017)

Aunque el enfoque analítico más simple es aplicar movimientos de superficie de campo libre, esto puede ser significativamente conservador según el tipo y la profundidad de la cimentación, ya que los movimientos de entrada efectivos del suelo al

nivel de la cimentación pueden reducirse significativamente en comparación con los movimientos de superficie. Por lo tanto, ignorar la interacción suelo-estructura es típicamente conservador para edificios típicos, pero se recomienda considerarlo para edificios con empotramiento significativo. Alternativamente, el movimiento de entrada se puede modificar para tener en cuenta los efectos del movimiento de entrada de base, que puede resultar en una reducción en el espectro de respuesta del objetivo, principalmente en el rango de alta frecuencia. Esto podría conducir a una modificación del enfoque de selección del movimiento del suelo y requerirá que el modelo tenga en cuenta explícitamente la interacción suelo-estructura a través del modelado no lineal de las suspensiones del suelo. Además, los efectos de inercia se pueden tener en cuenta al incluir componentes de cimentación en el modelo estructural.

Para poder efectuar el análisis no lineal tiempo – historia se debe discretizar el rango de tiempo de duración del movimiento del sismo en una serie de pasos discretos que son ejecutados de manera secuenciada y que por ende va a generar una respuesta estructural asociada delta de tiempo; de igual forma se debe discretizar el dominio espacial mediante la discretización de la estructura con el uso de una malla de elementos finitos que aproximen la geometría como las propiedades de la estructura. Estas necesidades de discretizaciones a la larga van a acumular errores comunes y que van a seguir acumulándose progresivamente vaya transcurriendo el análisis de tiempo. Y debido a la naturaleza de los errores que se generan en estos procesos, es complejo el poder determinar que existan errores; siendo esto más complejo ya que de por sí es difícil estimar la respuesta anticipada por medio de cálculos aproximados. Por lo tanto, es importante ser consciente de las posibles causas de los errores tanto espaciales como temporales en el análisis del historial de respuestas no lineal.

2.4Análisis Dinámico Incremental

Este segundo tipo de análisis dinámico no lineal es un método de análisis paramétrico el cual tiene como objetivo el estimar a mayor detalle el desempeño estructural bajo cargas sísmicas. Este análisis implica que el modelo realizado debe ser sometido a al menos un registro sísmico, el cual debe ser escalado en múltiples ocasiones para poder representar diversos niveles de intensidad sísmica con el fin de obtener curvas de respuestas sísmicas parametrizada frente al nivel de intensidad. Se examinan las propiedades de la curva IDA tanto para un solo grado de libertad (SDOF) como para múltiples grados de libertad (MDOF). (Vamvatsikos & Cornell, 2002)

Este método surgió debido al aumento de la capacidad de procesamiento de las computadoras, siendo este el último nivel de progreso en el análisis de estructuras ya que ha pasado de un análisis estático elástico al análisis dinámico elástico, estático no lineal y finalmente dinámico no lineal. Por analogía con el paso de un análisis estático único al empujón estático incremental, se llega a la extensión de un análisis histórico-temporal único a uno incremental, donde se escala la "carga" sísmica. Recientemente, también ha sido adoptado el Análisis Dinámico Incremental (IDA) y establecido como el método de vanguardia para determinar la capacidad de colapso global. El estudio IDA es ahora un método multipropósito y ampliamente aplicable y sus objetivos incluyen la comprensión completa del rango de respuesta versus el rango de niveles potenciales de un registro de movimiento del suelo.

También permite una mejor percepción de las implicaciones estructurales de niveles de movimiento del suelo más raros / más severos. Produce una mejor comprensión de los cambios en la naturaleza de la respuesta estructural a medida que aumenta la intensidad del movimiento del suelo que son basado en cambios en los patrones de

deformación máxima con la altura, en el conjunto de la degradación de la rigidez y resistencia, sus patrones y magnitudes.

También conocido simplemente como Análisis Dinámico Incremental (IDA), involucra una serie de ejecuciones dinámicas no lineales realizadas bajo imágenes escaladas de un acelerograma, cuyos IMs son, idealmente, seleccionados para cubrir todo el rango de elástico a no lineal y finalmente al colapso de la estructura. El propósito es registrar DM del modelo estructural en cada nivel IM del movimiento del suelo a escala, los valores de respuesta resultantes a menudo se trazan frente al nivel de intensidad como curvas continuas.

Capítulo 3: Códigos MATLAB

3.1OpenSees

Para poder entender la importancia de la generación del código de MATLAB para complementar la interfaz gráfica generada con la aplicación, es indispensable entender acerca del programa, así como en contexto en el cual surge. Históricamente, en temas ingenieriles relacionados con los terremotos no ha contado con la variedad de maneras o alternativas computacionales de realizar los procedimientos necesarios para los estudios pertinentes.

Recién por las décadas de los 70 y 80 fueron apareciendo nuevas herramientas informáticas relativamente básicas y basadas en aproximaciones funcionales para la época. Pero que contaban con la necesidad de requerir diferentes códigos para poder analizar y procesar diferentes necesidades correspondientes a la ingeniería sísmica en lo que compete, que van desde el análisis de peligros sísmicos, o sobre el comportamiento geotécnico del sitio, analizando la interacción suelo-estructura, y hasta realizando el análisis estructural de las mismas. Por lo cual la necesidad de obtener una herramienta

que pueda integrar todos los problemas mencionados en una simulación representaría una optimización y mejora en la ingeniería en cuestión.

Siendo por esta razón la creación del software OpenSees (Open System for Earthquake Engineering Simulation), el cual consiste básicamente en un código abierto que se encuentra orientado a objetos. Y más específicamente es una colección de módulos para facilitar la implementación de modelos y procedimientos de simulación para la ingeniería sísmica estructural y geotécnica. En donde se han desarrollado interfaces de software que permite la integración de modelos de estructuras y suelos para determinar posibles problemas o analizar resultados para sacar conclusiones sobre los temas en cuestión. Además, OpenSees está diseñado para aprovechar los últimos desarrollos en bases de datos, métodos de confiabilidad, visualización científica e informática de alta gama. (Pacific Earthquake Engineering Research Center, 2021)

OpenSees en otras palabras no es más que una plataforma informática aplicada en el desarrollo de simulación relacionadas con comportamientos en los elementos incidentes en los sistemas estructuras relacionados con sismos. La cual se encuentra complementada con la capacidad avanzada de modelación y análisis no lineal de los sistemas en cuestión, el cual se basa en modelos para los distintos elementos con sus correspondientes materiales y que encuentra la solución mediante diversos algoritmos que se basan en la teoría de análisis de los elementos finitos. (CEINCI, s.f)

En función de su mecanismo de análisis por elementos finitos representa la necesidad de generar la modelación de las estructuras mediante la lógica de clasificar a los componentes de la estructura en dos de los objetos principales para el análisis estructural como son los nodos y los elementos con el fin de poder establecer las restricciones nodales pertinentes y poder establecer las acciones de cargas a realizar.

Sobre la forma de realización y ejecución de los modelados pertinentes en el software, es realizada mediante una fuente abierta; lo que representa la posibilidad de ir constantemente actualizando las bases del software en función de las necesidades que se vayan presentando. Siendo por esta razón que como soporte para los comandos de OpenSees se implementa el lenguaje de herramientas de comando o Tool Command Language (TCL); y que se implementan en el software para la definición de la geometría del problema, estados de carga, formulación y solución. (CEINCI, s.f)

El TCL al ser un lenguaje de fácil aprendizaje, se lo usa en el desarrollo de todo tipo de programaciones. El cual puede ser fácilmente modificado de manera dinámica al ser un lenguaje interpretado; con la facultad de redefinir los comandos o sobrescribirlos de manera dinámica. Lo más importante de esto es que es un lenguaje multiplataforma, que puede ser interpretado y ejecutado por los diversos sistemas operativos. (CEINCI, s.f)

En función de las características del software mencionadas, es que recae la importancia del trabajo en cuestión puesto que este automatiza la generación de archivos para la implementación del software para conseguir los resultados del análisis no lineal y poder procesarlos para sacar conclusiones. Debido a la gran cantidad de condiciones que se deben definir del modelo a analizar para poder obtener los resultados, se desarrollaron algunos códigos distintos para poder ser integrados de manera dependiente del tipo de análisis mediante la aplicación de la interfaz. Por lo cual, para poder entender los códigos que se van a presentar a continuación, se presenta un pequeño diagrama de flujo donde se representa de manera general los códigos que se van a ejecutar automáticamente al seleccionar el tipo de análisis en la interfaz.

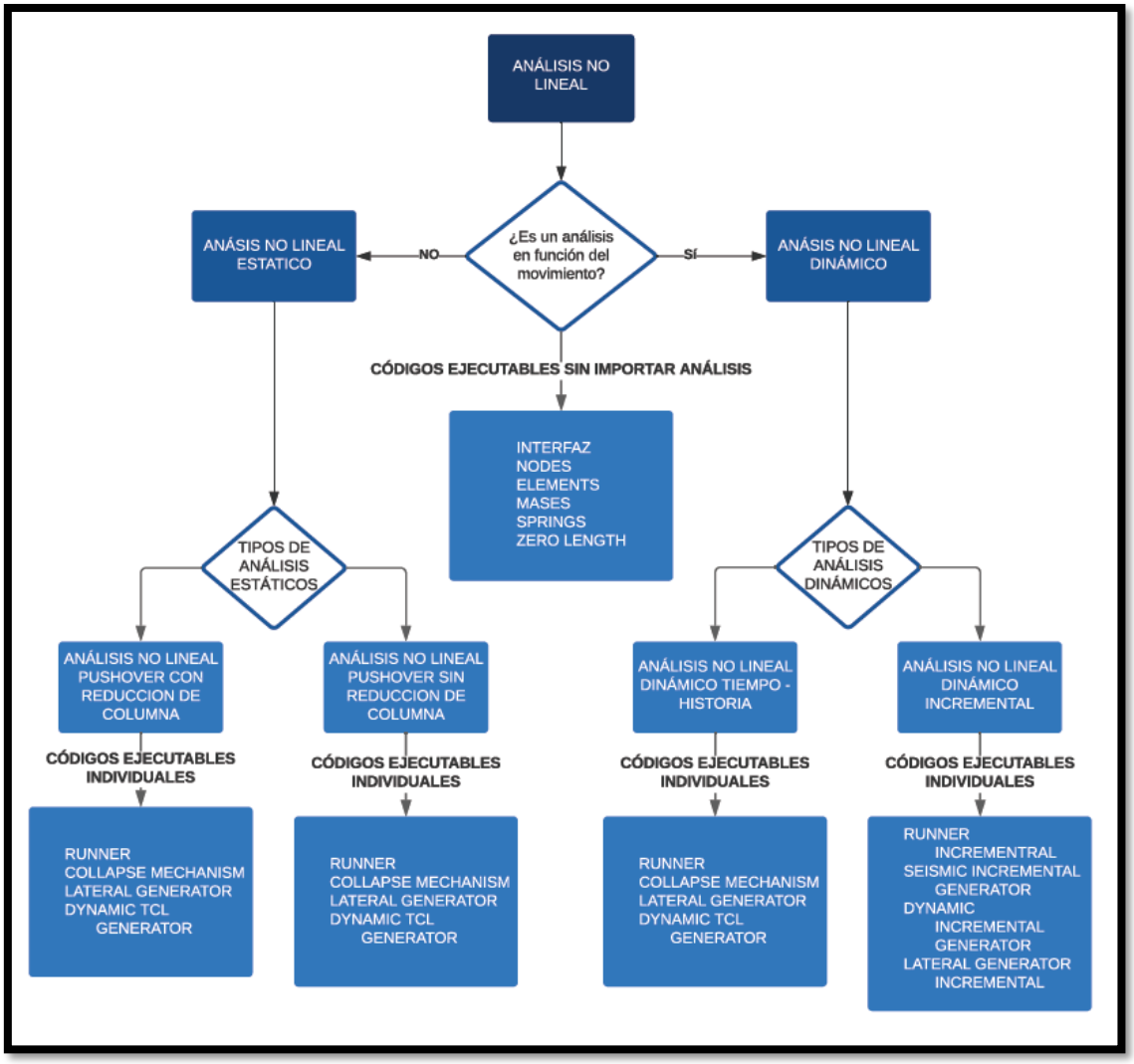


Ilustración 12: Diagrama de flujo de códigos a ejecutar en interfaz.

Fuente: Propia.

Finalmente, cabe recalcar que el código principal en el cual siempre se va a ejecutar de manera principal para poder ejecutar todo lo desarrollado es el código de la interfaz; por lo cual este código va a ser siempre del que van a depender los demás códigos.

3.2Generadores de Características

Explicación:

El Código generador de características para los elementos a analizar en la estructura es el código denominado Runner. Este código a su vez es dependiente de los archivos TCL generados por los códigos Nodes, Elements, Springs. Con los datos obtenidos de los archivos mencionados anteriormente, el propósito del código es el poder generar todas las propiedades geométricas correspondientes a cada uno de los elementos definidos anteriormente. En función de los valores que se ingresen en la interfaz con relación a los números de piso, se va a definir las secciones correspondientes a las vigas y columnas de cada nivel que son predefinidas para casos de investigación. Posteriormente, mediante el código se importan las propiedades de las secciones escogidas del catálogo de secciones del AISC como el área, la inercia, las dimensiones, entre otras propiedades necesarias de las estructuras de acero. Todos estos datos van a ser guardados en variables para cada elemento.

3.3Interfaz Gráfica

Explicación Código:

El presente código es el correspondiente a los comandos que ejecuta la interfaz creada tomando en cuenta los datos ingresados. El inicio del código sirve para poder leer y guardar los datos ingresados en la interfaz de los valores de la altura de la estructura, y su número de pisos. Posteriormente, en función del tipo de análisis que se va a realizar se ejecuta las subrutinas especificadas en cada uno de los 5 tipos de casos en donde el caso 1 es el análisis gravitacional, el tipo 2 es el análisis estático pushover con reducción de columnas, el tercer tipo es el análisis estático pushover sin la reducción de columnas, el tipo 4 es el análisis dinámico no lineal en función del tiempo – historia y finalmente el ultimo tipo es el análisis dinámico no lineal incremental. Adicionalmente el código en función de los tipos mencionados va a correr los gráficos y

tablas que se deben mostrar con los resultados obtenidos en el OpenSees para poder ser interpretados.

3.4 Análisis No Lineal

Para poder realizar los diversos procesos de análisis no lineal mediante el uso de la interfaz es primordial el explicar el mecanismo en el cual se ejecutan las modalidades. Por lo cual para poder realizar esto, el código divide en 5 posibles tipos de respuestas en función de lo seleccionado en la interfaz, lo que va a generar que se ejecuten cada una de las 5 subrutinas especificadas para cada caso. Posteriormente dentro de cada subcondición se llaman a los distintos códigos especificados en el diagrama de flujo; los cuales contienen todas las ecuaciones y comandos que el programa de OpenSees debe interpretar y ejecutar con los datos de la estructura previamente generados y guardados mediante los archivos TCL guardados en la carpeta principal. Habiéndose ejecutado los análisis correspondientes en OpenSees los resultados de respuesta son tomados por la interfaz y dependiendo del análisis realizado representa los resultados de manera gráfica en la interfaz tal como es el caso de la curva de Pushover.

Conclusiones

Concluyendo, los dos componentes principales de este trabajo fueron desarrollados con el fin de facilitar y optimizar el tiempo en que un ingeniero realiza tanto los procesos de diseño de marcos de acero arriostrados especiales, así como realizar el análisis no lineal de la mano de la aplicación de OpenSees. Si bien esta es la base para poder continuar con investigaciones a futuro que definan a un tema de investigación más detallado; este trabajo va a permitir obtener los resultados de los análisis de una manera más limpia y fácil de interpretar por los usuarios.

Esta mayor facilidad para realizar un análisis no lineal es se mucha importancia y ayuda para el contexto en el que nos encontramos en especial en el país, ya que se estaría en gran medida abordando el mayor problema del contexto ecuatoriano que tiene que ver con la dificultad de realizar un análisis. Con este trabajo se espera incentivar a más ingenieros a realizar un análisis no lineal en las estructuras de interés en vez de solo limitarse a realizar el análisis estático para no verse en la obligación de hacer procesos más tediosos de análisis.

De igual forma es importante evidenciar en las conclusiones obtenidas la dificultad de completamente generalizar tanto el programa en Excel como la interfaz generada en MATLAB. Esto a que al ser el análisis y diseño de estructuras con muchas variables que tienen incidencia de por medio, es muy complejo el poder comprimir todas las variables de análisis como valores obtenidos en productos relativamente simples. Por lo cual es un trabajo que se encuentra completamente abierto a futuros desarrollos con el fin de poder seguir facilitando y optimizando el proceso de análisis para dar una mayor seguridad y facilidad en los usuarios de poder obtener resultados más precisos en sus análisis tomando en cuanto muchas variables en consideración.

En síntesis, este trabajo de titulación que ya se viene trabajando con anterioridad por el director, es un gran paso para poder mejorar la forma de realizar los análisis más complejos en las estructuras que va a ayudar a minimizar el margen de error que se asumen y esperan al momento de diseñar y posteriormente construir una estructura.

Referencias

Brandonisio, G., Toreno, M., Grande, E., Mele, E., & De Luca, A. (2012). Seismic design of concentric braced frames. *Journal of Constructional Steel Research, 78*, 22-37.

Bruneau, M., Chia-Ming, U., & Sabelli, R. (2011). *Ductile Design of Steel Structures.* McGraw-Hill Education.

California, S. E. (2020). *2018 IBC SEAOC Structural/Seismic Design Manual.* Sacramento: SEAOC.

CEINCI. (s.f). *Uso de Opensees.* Quito: Escuela Politécnica del Ejercito.

Chao, S.-H., & Goel, S. (2006). A Seismic Design Method for Steel Concentric Braced Frames for Enhanced Performance. *4th International Conference on Earthquake Engineering*.

Deierlein, G., Reinhorn, A., & Willford, M. (2010). Nonlinear Structural Analysis For Seismic Design. *NEHRP Seismic Design Technical Brief No.4*, 1-36.

Duran, A. (2017). *ESTADO DEL ARTE DE ARRIOSTRAMIENTOS EN ESTRUCTURAS DE.* Santiago de Chile: Universidad de Chile.

INIFED. (2015). *Diseño de Estucturas de Acero: Normas y especificaciones para estudios, proyectos, construcción e instalaciones* (Vol. IV). Ciudad de México: Secretaría de Educación Pública.

Loges, S. (2017). Importancia de las Conexiones en el Comportamiento Sismirresistente de Edificaciones Aporticadas de Acero Estructural. *CONVESIS.* Caracas.

Massachussets Institute of Technology. (2010). *Introduction to Nonlinear Analysis.*

McCormac, J., & Csernak, S. (2012). *Structural Steel Design.* New Jersey: Pearson Education.

Ministerio de Desarrollo Urbano y Vivienda. (2015). *Norma Ecuatoriana de la Construcción NEC.* Quito: MIDUVI.

Momenzadeh, S. (2017). *Seismic design study of concentrically braced frames with and without buckling-controlled braces.* Ames: Iowa State University.

National Institute of Standards and Technology. (2017). *Guidelines for Nonlinear Structural Analysis for Design of Buildings.* Redwood City: NIST.

National Institute of Standards and Technology. (2017). *Recommended Modeling Parameters and Aceptance Criteria for Non Linear Analysis in Support of Seismic Evaluation, Retrofit, and Design.* Redwood City: NIST.

Oviedo, J., & Duque, M. (2006). Sistemas de Control de Respuesta Sísmica en Edificaciones. *EIA*, 105-120.

Pacific Earthquake Engineering Research Center. (2021). *PEER*. (B. University of California, Editor) Obtenido de OpenSees - The Open System for Earthquake Engineering Simulation: https://apps.peer.berkeley.edu/products/opensees.html

Rofooei, F., Attari, N., Rasekh, A., & Shodja, A. (2006). Comparison of Static and Dynamic Pushover Analysis in Assessment of theTarget Displacement. *International Journal of Civil Engineering*, 212-225.

Sabelli, R., Roeder, C., & Hajjar, J. (2013). Seismic Design of Steel Special Concentrically Braced Frame Systems. *NEHRP Seismic Design Technical Brief No. 8* .

Salomon, C., Johnson, J., & Malhas, F. (2020). *Steel Structures: Design and Behavior.* New Jersey: Pearsons.

Vamvatsikos, D., & Cornell, A. (2002). *Incremental Dynamic Analysis.* California.

Wen, R., Seker, O., Akbas, B., & Shen, J. (2016). Designs of Special Concentrically Braced Frame Using AISC 341-05 and AISC 341-10. *Practice Periodical on Structural Design and Construction*.

Yang, T., Sheikh, H., & Tobber, L. (2019). Influence of the Brace Configurations on the Seismic Performance of Steel Concentrically Braced Frames. *Innovative Methodologies for Resilient Buildings and CIties*.

Anexo A: Código Macro Excel Distribución Fuerzas

```
Private Sub DistribucionFuerzas_Click()
    Dim Rng As Range
    Dim WorkRng As Range
    On Error Resume Next
    xTitleId = "RANGO APLICACION"
    Set WorkRng = Range("b25", Range("b50").End(xlUp))
    Set WorkRng = WorkRng.Columns(1)
    xLastRow = WorkRng.Rows.Count
    Application.ScreenUpdating = False
    For xRowIndex = xLastRow To 1 Step -1
      Set Rng = WorkRng.Range("B" & xRowIndex)
      If Rng.Value < "C18" Then
      If Rng.Value > "0" Then
        ActiveSheet.Range("B25").Select
        k = Range("c18").Value
        For i = 1 To k - 1
        ActiveCell.Value = i
        ActiveCell.Offset(1, 0).Activate
        Rng.Offset(1, 0).EntireRow.Insert Shift:=xlDown
        Next i

      End If
      End If
    Next
```

```
 ActiveSheet.Range("B25").Select
        k = Range("c18").Value
        For i = 1 To k
        ActiveCell.Value = i
        ActiveCell.Offset(1, 0).Activate
        Next i


Range("C26").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=R[-1]C"
Range("D26").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=R[-1]C"
Range("E26").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=R[-1]C+RC[-1]"
Range("F26").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=RC[-1]*R22C3"
Range("G26").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=RC[-4]*RC[-1]"
Range("H26").Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=RC[-1]/R23C7"
```

```
    Range("I26").Select
    Application.CutCopyMode = False
    ActiveCell.FormulaR1C1 = "=RC[-1]*R12C5"
    Range("C26:I26").Select
    Selection.AutoFill Destination:=Range("C26:I" & Range("B" &
Rows.Count).End(xlUp).Row)
    Application.ScreenUpdating = True
End Sub
```

Anexo B: Código MATLAB Principal Generación de Características

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

type
aaa=pwd;
cd(aaa)

%Structural configuration:

NumS;
bay;
h1;
hi;
Ry;
type;

%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

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','W27X
94','W24X84','W24X84'}';
```

```
    extcol_sizes = {'W24X207', 'W24X207', 'W24X207',
'W24X162', 'W24X162', 'W24X146', 'W24X146', 'W24X131',
'W24X131', 'W24X131', 'W24X131', 'W24X84'}';
    incol_sizes =  {'W24X207', 'W24X207', 'W24X207',
'W24X207', 'W24X207', 'W24X176', 'W24X176', 'W24X162',
'W24X162', 'W24X131', 'W24X131', 'W24X94'}';

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

    case 20
    %TWENTY STORY BUILDING:

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

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

end
%Material properties:

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

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

AISC_ID = fopen('AISC.txt');
AISC_data = textscan(AISC_ID, '%s %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f %f %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=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)+2.9
481e-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,b
eam_A, beam_Ix);
ZeroLength(NumS);
Springs(NumS,Fy,extcol_d,extcol_tw,extcol_bf,extcol_tf
,incol_d,incol_tw,incol_bf,incol_tf,beam_d,ext_col_the
```

```
ta_p,ext_col_theta_pc,ext_col_lambda,Ke_ext_col,My_ext
_col,ass_ext_col,int_col_theta_p,int_col_theta_pc,int_
col_lambda,Ke_int_col,My_int_col,ass_int_col,My_RBS,be
am_theta_p,beam_theta_pc,beam_lambda,Ke_beam,ass_beam,
ext_dblplate,int_dblplate,reduction_capacity_ext,reduc
tion_capacity_int);
masses(NumS);
end
```

Anexo C: Código MATLAB Interfaz Gráfica

```
function varargout = Interfaz(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton,
...
                   'gui_OpeningFcn',
@Interfaz_OpeningFcn, ...
                   'gui_OutputFcn',
@Interfaz_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Interfaz is made visible.
function Interfaz_OpeningFcn(hObject, ~, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

```
% varargin   command line arguments to Interfaz (see
VARARGIN)

% Choose default command line output for Interfaz
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes Interfaz wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the
command line.
function varargout = Interfaz_OutputFcn(~, ~, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles
structure
varargout{1} = handles.output;


function inputstories_Callback(hObject, ~, ~)
% hObject    handle to inputstories (see GCBO)
NumS=str2double(get(hObject,'String'));
hObject.UserData=NumS;

% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
inputstories as text
%        str2double(get(hObject,'String')) returns
contents of inputstories as a double
```

```
% --- Executes during object creation, after setting
all properties.
function inputstories_CreateFcn(hObject, ~, ~)
% hObject    handle to inputstories (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function inputbay_Callback(hObject, ~, ~)
% hObject    handle to inputbay (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
bay=str2double(get(hObject,'String'));
% handles    structure with handles and user data (see
GUIDATA)
hObject.UserData=bay;
% Hints: get(hObject,'String') returns contents of
inputbay as text
%        str2double(get(hObject,'String')) returns
contents of inputbay as a double

% --- Executes during object creation, after setting
all properties.
function inputbay_CreateFcn(hObject, ~, ~)
% hObject    handle to inputbay (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function inputh1_Callback(hObject, ~, ~)
% hObject    handle to inputh1 (see GCBO)
h1=str2double(get(hObject,'String'));
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
hObject.UserData=h1;
% Hints: get(hObject,'String') returns contents of
inputh1 as text
%        str2double(get(hObject,'String')) returns
contents of inputh1 as a double


% --- Executes during object creation, after setting
all properties.
function inputh1_CreateFcn(hObject, ~, ~)
% hObject    handle to inputh1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function inputhi_Callback(hObject, ~, ~)
% hObject    handle to inputhi (see GCBO)
hi=str2double(get(hObject,'String'));
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
hObject.UserData=hi;
% Hints: get(hObject,'String') returns contents of
inputhi as text
%        str2double(get(hObject,'String')) returns
contents of inputhi as a double
```

```
% --- Executes during object creation, after setting
all properties.
function inputhi_CreateFcn(hObject, ~, ~)
% hObject    handle to inputhi (see GCBO)

% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function InputRy_Callback(hObject, ~, ~)
% hObject    handle to InputRy (see GCBO)
Ry=str2double(get(hObject,'String'));
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
hObject.UserData=Ry;
% Hints: get(hObject,'String') returns contents of
InputRy as text
%        str2double(get(hObject,'String')) returns
contents of InputRy as a double


% --- Executes during object creation, after setting
all properties.
function InputRy_CreateFcn(hObject, ~, ~)
% hObject    handle to InputRy (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

% --- Executes on selection change in TypesList.
function TypesList_Callback(~, ~, ~)

% hObject    handle to TypesList (see GCBO)



% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: contents = cellstr(get(hObject,'String'))
returns TypesList contents as cell array
%        contents{get(hObject,'Value')} returns
selected item from TypesList


% --- Executes during object creation, after setting
all properties.
function TypesList_CreateFcn(hObject, ~, ~)
% hObject    handle to TypesList (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: listbox controls usually have a white
background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting
all properties.
function StructureGraph_CreateFcn(~, ~, ~)
% hObject    handle to StructureGraph (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called
```

```
% Hint: place code in OpeningFcn to populate
StructureGraph


% --- Executes on mouse press over axes background.
function StructureGraph_ButtonDownFcn(~, ~, ~)
% hObject    handle to StructureGraph (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% --- Executes on selection change in listbox2.



function listbox2_Callback(~, ~, ~)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: contents = cellstr(get(hObject,'String'))
returns listbox2 contents as cell array
%        contents{get(hObject,'Value')} returns
selected item from listbox2


% --- Executes during object creation, after setting
all properties.
function listbox2_CreateFcn(hObject, ~, ~)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: listbox controls usually have a white
background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in ButtonProcessData.
function ButtonProcessData_Callback(~, ~, handles)
% hObject    handle to ButtonProcessData (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
cd
('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTclFi
les\OpenSees3.2.2-x64.exe\bin');

%Structural configuration:

NumS=handles.inputstories.UserData;     %Number of
stories of the building
bay=handles.inputbay.UserData*12;   %Bay length
h1=handles.inputh1.UserData*12;    %Height of the
first story
hi=handles.inputhi.UserData*12;     %Height of the
rest of stories
Ry=handles.InputRy.UserData;
typegeneral=handles.TypesList.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

if typegeneral==1
    type=2;
    Runner(NumS, bay, h1, hi, Ry, type, typegeneral)
    GravityGenerator
    GravityGeneralGenerator

end

if typegeneral==2
    type=2;
    Runner(NumS, bay, h1, hi, Ry, type, typegeneral)
```

```
    dynamicTclGenerator(NumS)
    LateralGenerator


end

if typegeneral==3
    type=1;
    Runner(NumS, bay, h1, hi, Ry, type, typegeneral)
    dynamicTclGenerator(NumS)
    LateralGenerator

end

if typegeneral==4
    type=1;
    Runner(NumS, bay, h1, hi, Ry, type, typegeneral)
    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(Xbay1(l:l+1),Y(l:l+1));
plot(Xbay2(l:l+1),Y(l:l+1));
plot(Xbay3(l:l+1),Y(l:l+1));
hold 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(Xc1b1,Yg);
plot(Xc2b1,Yg);
plot(Xc3b2,Yg);
plot(Xc4b3,Yg);
xlim([-1 3*bay+1]);
ylim([0 h1+(NumS-1)*hi+1]);




% --- Executes during object creation, after setting
all properties.
function ResponseGraph_CreateFcn(hObject, eventdata,
handles)
% hObject    handle to ResponseGraph (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called
```

```
% Hint: place code in OpeningFcn to populate
ResponseGraph

function CollapseMechanismGraph_CreateFcn(hObject,
eventdata, handles)



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(~, ~, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
typegeneral=handles.TypesList.Value;

if typegeneral==1

    Status=system('opensees.exe gravity.tcl');

end




if typegeneral==2

    Status=system('opensees.exe lateral.tcl');


    cd
('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTclFi
les\OpenSees3.2.2-x64.exe\bin\Concentrated-PanelZone-
Pushover-Output')

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=handles.inputstories.UserData;      %Number of
stories of the building
bay=handles.inputbay.UserData*12;    %Bay length
h1=handles.inputh1.UserData*12;     %Height of the
first story
hi=handles.inputhi.UserData*12;      %Height of the
rest of stories
Ry=handles.InputRy.UserData;
typegeneral=handles.TypesList.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
('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTclFi
les\OpenSees3.2.2-x64.exe\bin')

distancetospring=14.9;

% matrix=importdata('Nodes.tcl',' ',6);

fileId = fopen("Nodes.tcl");
C = textscan(fileId, "%s %d %f %f", 'CommentStyle',
'#');
nodes = cell2mat(C(:, 3:4));

cd('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTcl
Files\OpenSees3.2.2-x64.exe\bin\Concentrated-
PanelZone-Pushover-Output')



% A=matrix.data
% nodes=A(:,2:3);
Push_Over=load('Disp_Roof.out');
level=200;
scale=1.0;

base_rot=load('MRFbase-Rot.out');
base_mom=load('MRFbase-Mom.out');


col_rot_1=load('MRFcol-Rot-1-1.out');
col_mom_1=load('MRFcol-Mom-1-1.out');
col_rot_2=load('MRFcol-Rot-2-1.out');
col_mom_2=load('MRFcol-Mom-2-1.out');
col_rot_3=load('MRFcol-Rot-3-1.out');
col_mom_3=load('MRFcol-Mom-3-1.out');
```

```
Col=[col_rot_1 col_mom_1 col_rot_2 col_mom_2 col_rot_3
col_mom_3];


for i=2:num+1

    BeamsMom(i-1,1)={['MRFbeam-Mom-' num2str(i) '-
1.out']};

end


for i=2:num+1

    BeamsRot(i-1,1)={['MRFbeam-Rot-' num2str(i) '-
1.out']};

end


for i=1:num
   Var=load(BeamsMom{i});
   n_mom = -Var(1:level,4);
   beam_n_mom(1:level,i) = n_mom;
end


for i=1:num
   Var=load(BeamsRot{i});
   n_rot = Var(1:level,2);
   beam_n_rot(1:level,i) = n_rot;
end


for k=1:num

    for i=1:size(nodes,1)

         if and(nodes(i,2)>=(h1+hi*(k-1)-
distancetospring-10),nodes(i,2)<=(h1+hi*(k-1)-
distancetospring+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
        end
        if and(nodes(i,2)>=(h1+hi*(k-1)-
10),nodes(i,2)<=(h1+hi*(k-1)+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
```

```
        end
        if and(nodes(i,2)>=(h1+hi*(k-
1)+distancetospring-10),nodes(i,2)<=(h1+hi*(k-
1)+distancetospring+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
        end
        if and(nodes(i,2)>=(h1+hi*(k-1)+hi/2-
10),nodes(i,2)<=(h1+hi*(k-1)+hi/2+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
        end
    end


end


%Plastic Hinges:
if level==987
    pls_1=load('hinges_1.txt');
elseif level==381
    pls_1=load('hinges_2.txt');
end

%Plot all the nodes:
axes(handles.CollapseMechanismGraph);

figure('Name','Nodes','NumberTitle','off','Color','whi
te')

axes(handles.CollapseMechanismGraph);

plot(nodes(:,1),nodes(:,2),'o','markersize',4,'color',
'black')
hold on

%
plot(960,0,'O','MarkerFaceColor','red','MarkerSize',10
)
% hold on
if level==987 || level==381
    for i=1:4
        axes(handles.CollapseMechanismGraph);

plot(nodes(i,1),nodes(1,2),'O','MarkerFaceColor','red'
,'MarkerSize',22)
    axes(handles.CollapseMechanismGraph);
```

```
plot(nodes(i,1),40,'O','MarkerFaceColor','red','Marker
Size',22)
    end
    hold on
    axes(handles.CollapseMechanismGraph);

plot(pls_1(:,1),pls_1(:,2),'O','MarkerFaceColor','red'
,'MarkerSize',22)
end
hold off
axis 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;
k=NumS+1;
kk=NumS;
aa=0;
for i=3:k
   yc1(i)=yc1(2)+156.1*(i-2);
   yc2(i)=yc1(i)+126.1;
end

for j=1:5
    axes(handles.CollapseMechanismGraph);
    line([xc1+aa,xc2+aa+Push_Over(level,9)*scale],
[yc1(1), yc2(1)],'color','black','LineWidth',2.9)
    aa=aa+240;
end

for j=1:5
    for i=2:k
        axes(handles.CollapseMechanismGraph);

line([xc1+a+Push_Over(level,k)*scale,xc2+a+Push_Over(l
evel,kk)*scale], [yc1(i),
yc2(i)],'color','black','LineWidth',2.9)
        k=k-1;
        kk=kk-1;
```

```
    end
    a=a+240;

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=2:8
    yb1(i)=yb1(1)+156*(i-1);
    yb2(i)=yb1(i);
end
k=9;
for i=1:8
    for j=1:4
        axes(handles.CollapseMechanismGraph);

line([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=3:8
   yp1(i)=yp1(2)+156*(i-2);
```

```
end

yp2=yp1;
k=9;
for i=1:8
    for j=1:4
        axes(handles.CollapseMechanismGraph);
        line([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=9;
for i=1:8
    for j=1:4
        axes(handles.CollapseMechanismGraph);
        line([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=9;
for i=1:8
    for j=1:4
        axes(handles.CollapseMechanismGraph);
        line([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=9;
for i=1:8
    for j=1:4
        axes(handles.CollapseMechanismGraph);
        line([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

if typegeneral==3

    Status=system('opensees.exe lateral.tcl');
  cd
('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTclFi
les\OpenSees3.2.2-x64.exe\bin\Concentrated-PanelZone-
Pushover-Output')
    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=handles.inputstories.UserData;     %Number of
stories of the building
bay=handles.inputbay.UserData*12;    %Bay length
h1=handles.inputh1.UserData*12;    %Height of the
first story
hi=handles.inputhi.UserData*12;     %Height of the
rest of stories
Ry=handles.InputRy.UserData;
typegeneral=handles.TypesList.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
('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTclFi
les\OpenSees3.2.2-x64.exe\bin')

axes(handles.CollapseMechanismGraph);
cd('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTcl
Files\OpenSees3.2.2-x64.exe\bin')



distancetospring=14.9;

% matrix=importdata('Nodes.tcl',' ',6);

fileId = fopen("Nodes.tcl");
C = textscan(fileId, "%s %d %f %f", 'CommentStyle',
'#');
nodes = cell2mat(C(:, 3:4));
```

```
cd('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTcl
Files\OpenSees3.2.2-x64.exe\bin\Concentrated-
PanelZone-Pushover-Output')



% A=matrix.data
% nodes=A(:,2:3);
Push_Over=load('Disp_Roof.out');
level=200;
scale=1.0;

base_rot=load('MRFbase-Rot.out');
base_mom=load('MRFbase-Mom.out');


col_rot_1=load('MRFcol-Rot-1-1.out');
col_mom_1=load('MRFcol-Mom-1-1.out');
col_rot_2=load('MRFcol-Rot-2-1.out');
col_mom_2=load('MRFcol-Mom-2-1.out');
col_rot_3=load('MRFcol-Rot-3-1.out');
col_mom_3=load('MRFcol-Mom-3-1.out');

Col=[col_rot_1 col_mom_1 col_rot_2 col_mom_2 col_rot_3
col_mom_3];


for i=2:num+1

    BeamsMom(i-1,1)={['MRFbeam-Mom-' num2str(i) '-
1.out']};

end


for i=2:num+1

    BeamsRot(i-1,1)={['MRFbeam-Rot-' num2str(i) '-
1.out']};

end


for i=1:num
   Var=load(BeamsMom{i});
   n_mom = -Var(1:level,4);
   beam_n_mom(1:level,i) = n_mom;
```

```
end


for i=1:num
   Var=load(BeamsRot{i});
   n_rot = Var(1:level,2);
   beam_n_rot(1:level,i) = n_rot;
end


for k=1:num

    for i=1:size(nodes,1)

         if and(nodes(i,2)>=(h1+hi*(k-1)-
distancetospring-10),nodes(i,2)<=(h1+hi*(k-1)-
distancetospring+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
        end
        if and(nodes(i,2)>=(h1+hi*(k-1)-
10),nodes(i,2)<=(h1+hi*(k-1)+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
        end
        if and(nodes(i,2)>=(h1+hi*(k-
1)+distancetospring-10),nodes(i,2)<=(h1+hi*(k-
1)+distancetospring+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
        end
        if and(nodes(i,2)>=(h1+hi*(k-1)+hi/2-
10),nodes(i,2)<=(h1+hi*(k-1)+hi/2+10))
       nodes(i,1)=nodes(i,1)+Push_Over(level,num+2-
k)*scale;
        end
    end


end


%Plastic Hinges:
if level==987
    pls_1=load('hinges_1.txt');
elseif level==381
    pls_1=load('hinges_2.txt');
end
```

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

%
plot(960,0,'O','MarkerFaceColor','red','MarkerSize',10
)
% hold on
if level==987 || level==381
    for i=1:4

plot(nodes(i,1),nodes(1,2),'O','MarkerFaceColor','red'
,'MarkerSize',22)

plot(nodes(i,1),40,'O','MarkerFaceColor','red','Marker
Size',22)
    end
    hold on

plot(pls_1(:,1),pls_1(:,2),'O','MarkerFaceColor','red'
,'MarkerSize',22)
end
hold off
axis 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=3:9
   yc1(i)=yc1(2)+156.1*(i-2);
   yc2(i)=yc1(i)+126.1;
end
k=9;
kk=8;
aa=0;
for j=1:5
```

```
    line([xc1+aa,xc2+aa+Push_Over(level,9)*scale],
[yc1(1), yc2(1)],'color','black','LineWidth',2.9)
    aa=aa+240;
end

for j=1:5
    for i=2:8

line([xc1+a+Push_Over(level,k)*scale,xc2+a+Push_Over(l
evel,kk)*scale], [yc1(i),
yc2(i)],'color','black','LineWidth',2.9)
        k=k-1;
        kk=kk-1;

    end
    a=a+240;
    k=9;
    kk=8;
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=2:8
    yb1(i)=yb1(1)+156*(i-1);
    yb2(i)=yb1(i);
end
k=9;
for i=1:8
    for j=1:4

line([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=3:8
   yp1(i)=yp1(2)+156*(i-2);
end

yp2=yp1;
k=9;
for i=1:8
    for j=1:4
        line([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=9;
for i=1:8
    for j=1:4
        line([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=9;
for i=1:8
    for j=1:4
        line([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=9;
```

```
for i=1:8
    for j=1:4
        line([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

if typegeneral==4

    Status=system('opensees.exe lateral.tcl');


    cd
('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTclFi
les\OpenSees3.2.2-x64.exe\bin\Results')

%Load all the demands:
numq=10; %number of GMs
num_pisos=8;
for k = 1:numq
  myfilename1 = sprintf('Accel%d.out', k); %this
command creates a string with the name of each output
file
  myfilename2 = sprintf('Drift%d.out', k);
  A{k} = importdata(myfilename1); %this matrix
accumlates the data from accelerations
  Drift{k} = importdata(myfilename2); %this matrix
accumlates the data from drifts
end

for i=1:numq
    for j=2:num_pisos+1
        Floor_acc(i,j-1)=max(abs(A{i}(:,j)));
        Max_Drifts(i,j-1)=max(abs(Drift{i}(:,j)));
    end

end



    set(handles.Tabla,'Data',A)

end
```

```
% --- Executes during object creation, after setting
all properties.
function Tabla_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tabla (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called
```

Anexo D: Código MATLAB Generación Nodos Estructura

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

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
```

Anexo E: Código MATLAB Generación Masas en Estructura

```
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
```

Anexo F: Código MATLAB Generación Masas en Estructura

```
function
Elements(nf,NumS,extcol_A,incol_A,extcol_Ix,incol_Ix,b
eam_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
```

Anexo G: Código MATLAB para Análisis Gravitacional

```
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 n   10;',' ');

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','  ',' recorder Node -file
$dataDir/eigenvalues.out -node');

k=2;
for i=2: NumS+1

fprintf(fileID, '%s%d',' 1',k,'005 ') ;
k=k+1;
```

```
end

fprintf(fileID, '%s','  ','-dof 1 "eigen 1"','','');

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

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

fprintf(fileID, '%s\r\n','  set nEigenI 1;
# mode i = 1');
fprintf(fileID, '%s\r\n','  set nEigenJ 2;
# mode j = 2');
fprintf(fileID, '%s\r\n','  set nEigenK 3;
# mode k = 3');
fprintf(fileID, '%s\r\n','  set nEigenL 4;
# mode l = 4');
fprintf(fileID, '%s\r\n','  set nEigenM 5;
# mode m = 5');
fprintf(fileID, '%s\r\n','  set nEigenO 6;
# mode o = 6');
fprintf(fileID, '%s\r\n','  set nEigenP 7;
# mode p = 7');
fprintf(fileID, '%s\r\n','  set nEigenQ 8;
# mode q = 8','','','');


fprintf(fileID, '%s\r\n','  set lambdaN [eigen  -
fullGenLapack  [expr $nEigenQ]];           #
eigenvalue analysis for nEigenQ modes');
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','','');


fprintf(fileID, '%s\r\n','  set w1 [expr
pow($lambdaI,0.5)];            # w1 (1st mode circular
frequency)','   set w2 [expr pow($lambdaJ,0.5)];
# w2 (2nd mode circular frequency)','   set w3 [expr
pow($lambdaK,0.5)];            # w3 (3st mode circular
frequency)','   set w4 [expr pow($lambdaL,0.5)];
# w4 (4nd mode circular frequency)','   set w5 [expr
pow($lambdaM,0.5)];            # w5 (5st mode circular
frequency)','   set w6 [expr pow($lambdaO,0.5)];
# w6 (6nd mode circular frequency)','   set w7 [expr
pow($lambdaP,0.5)];            # w7 (7st mode circular
frequency)','   set w8 [expr pow($lambdaQ,0.5)];
# w8 (8nd mode circular frequency)','','','');

k=1;

for i=1:8

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

Anexo H: Código MATLAB para Análisis Dinámico Incremental

```
cd('C:\Users\DELL\OneDrive\Documents\TESIS\GenerateTcl
Files\OpenSees3.2.2-x64.exe\bin');

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


fprintf(fileID, '%s\r\n', '# Start ground motion
iteration:','');

fprintf(fileID, '%s%d', 'set GMinput [open
"recordSismo',i,'.txt" r];');

fprintf(fileID, '%s\r\n', '','','set GMs [split [read
$GMinput] "\n"]; # each line contains 4 elements');
fprintf(fileID, '%s\r\n','  #set GmsP [lrange $GMs 0
end-1];');
fprintf(fileID, '%s\r\n','  close $GMinput;','');
fprintf(fileID, '%s\r\n','foreach GMrecord "$GMs"
{;');
fprintf(fileID, '%s\r\n','  wipe');
fprintf(fileID, '%s\r\n','  source
GravityGeneral.tcl');
fprintf(fileID, '%s\r\n','# display deformed shape:');
fprintf(fileID, '%s\r\n','#set ViewScale 10;
# amplify display of deformed shape');
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','# Rayleigh Damping');
fprintf(fileID, '%s\r\n','      # calculate damping
parameters');
fprintf(fileID, '%s\r\n','      set zeta 0.0250;
# percentage of critical damping');
fprintf(fileID, '%s\r\n','      set n 10;');
fprintf(fileID, '%s\r\n','      set a0 [expr
$zeta*2.0*$w1*$w3/($w1 + $w3)];    # mass damping
coefficient based on first and second modes');
fprintf(fileID, '%s\r\n','      set a1 [expr
$zeta*2.0/($w1 + $w3)];        # stiffness damping
coefficient based on first and second modes');
fprintf(fileID, '%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, '%s\r\n','      # assign damping to
frame beams and columns ');
fprintf(fileID, '%s\r\n','      # command: region
$regionID -eleRange $elementIDfirst $elementIDlast
rayleigh $alpha_mass $alpha_currentStiff
$alpha_initialStiff $alpha_committedStiff');
fprintf(fileID, '%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, '%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, '%s\r\n','  # define ground motion
parameters');
fprintf(fileID, '%s\r\n','      set patternID 1;
# load pattern ID');
fprintf(fileID, '%s\r\n','      set GMdirection 1;
# ground motion direction (1 = x)');
fprintf(fileID, '%s\r\n','      set j [lindex [split
$GMrecord] 0]; ');
fprintf(fileID, '%s\r\n','      set name [lindex
[split $GMrecord] 1];');
fprintf(fileID, '%s\r\n','      set NPTS [lindex
[split $GMrecord] 2];');
fprintf(fileID, '%s\r\n','      set deltat [lindex
[split $GMrecord] 3];');
fprintf(fileID, '%s\r\n','      set scalefactor
[lindex [split $GMrecord] 4];');
fprintf(fileID, '%s\r\n','      set GMtime [expr
$NPTS*$deltat + 0.00]; # total time of ground motion +
10 sec of free vibration');
fprintf(fileID, '%s\r\n','  # define the acceleration
series for the ground motion');

fprintf(fileID, '%s\r\n','  # syntax:  "Series -dt
$timestep_of_record -filePath
$filename_with_acc_history -factor
$scale_record_by_this_amount');
```

```
fprintf(fileID, '%s\r\n','      set accelSeries
"Series -dt $deltat -filePath $name -factor [expr
$scalefactor*$g]";');

fprintf(fileID, '%s\r\n','  # create load pattern:
apply acceleration to all fixed nodes with
UniformExcitation');

fprintf(fileID, '%s\r\n','      # command: pattern
UniformExcitation $patternID $GMdir -accel
$timeSeriesID ');

fprintf(fileID, '%s\r\n','      pattern
UniformExcitation $j $GMdirection -accel
$accelSeries;');

fprintf(fileID, '%s%d', '       set dataDir
IncrementalAnalysis/Sismo',i);


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

fprintf(fileID, '%s\r\n','      recorder Node -file
$dataDir/R$j.out -time -node 11 21 31 41 51 -dof 1 2 3
reaction;');
fprintf(fileID, '%s\r\n','      recorder Drift -file
$dataDir/Drift$j.out -time -iNode 11 12005 13005 14005
15005 16005 17005 18005 -jNode 12005 13005 14005 15005
16005 17005 18005 19005 -dof 1 -perpDirn 2;');
fprintf(fileID, '%s\r\n','      recorder Node -file
$dataDir/Accel$j.out -time -node 12005 13005 14005
15005 16005 17005 18005 19005 -dof 1 accel;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFbase-Connec-Mom$j.out -time -ele 1 2 3 4
force;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFbase-Connec-Rot$j.out -time -ele 1 2 3 4
deformation;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFcolbase-Mom$j.out -time -ele 3111 3211
3311 3411 force;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFcolbase-Rot$j.out -time -ele 3111 3211
3311 3411 deformation;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRcol1-Mom$j.out -time -ele 3112 3212 3312
3412 force;');
```

```
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRcol1-Rot$j.out -time -ele 3112 3212 3312
3412 deformation;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRcol2-Mom$j.out -time -ele 3121 3221 3321
3421 force;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRcol2-Rot$j.out -time -ele 3121 3221 3321
3421 deformation;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFbeam2-Mom$j.out -time -ele 4122 4221 4222
4321 4322 4421 force;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFbeam2-Rot$j.out -time -ele 4122 4221 4222
4321 4322 4421 deformation;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFbeam3-Mom$j.out -time -ele 4132 4231 4232
4331 4332 4431 force;');
fprintf(fileID, '%s\r\n','      recorder Element -file
$dataDir/MRFbeam3-Rot$j.out -time -ele 4132 4231 4232
4331 4332 4431 deformation;');
fprintf(fileID, '%s\r\n','      # define dynamic
analysis parameters');
fprintf(fileID, '%s\r\n','      set DtAnalysis 0.001;
# timestep of analysis');
fprintf(fileID, '%s\r\n','      wipeAnalysis
# destroy all components of the Analysis object, i.e.
any objects created with system, numbere');
fprintf(fileID, '%s\r\n','      constraints
Transformation; # 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 UmfPack;
# how to store and solve the system of equations in
the analysis');
fprintf(fileID, '%s\r\n','      test NormUnbalance
0.01 100;    # type of convergence criteria with
tolerance, max iterations');
fprintf(fileID, '%s\r\n','      set algorithmType
Newton; #ModifiedNewton ');
fprintf(fileID, '%s\r\n','      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 Newmark 0.5
0.25;    # uses Newmarks average acceleration method
to compute the time history');

fprintf(fileID, '%s\r\n','      analysis Transient;
# type of analysis: transient or static');
fprintf(fileID, '%s\r\n','      set NumSteps [expr
round(($GMtime + 0.0)/$DtAnalysis)]; # number of steps
in analysis');
fprintf(fileID, '%s\r\n','       puts "";');
fprintf(fileID, '%s\r\n','      puts "Beginning Ground
Motion Analysis";');
fprintf(fileID, '%s\r\n','      for {set i 1} {$i <=
$NumSteps} {incr i 1} {');
fprintf(fileID, '%s\r\n','          set ok [analyze 1
$DtAnalysis];         # actually perform analysis --
returns ok=0 if analysis was successful');
fprintf(fileID, '%s\r\n','          puts "GM step $i
out of $NumSteps: $ok";');
fprintf(fileID, '%s\r\n','         if {$ok != 0} {');
fprintf(fileID, '%s\r\n','              source
GM_convergence_loop.tcl;');
fprintf(fileID, '%s\r\n','          };');
fprintf(fileID, '%s\r\n','      }');
fprintf(fileID, '%s\r\n','      puts "End of Ground
Motion Analysis";');

fprintf(fileID, '%s\r\n','      # output time at end
of analysis');

fprintf(fileID, '%s\r\n','          set currentTime
[getTime];  # get current analysis time (after dynamic
analysis)');
fprintf(fileID, '%s\r\n','          puts "The current
time is: $currentTime";');



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