UNIVERSIDAD DE CHILE

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

DEPARTAMENTO DE INGENIERIA CIVIL

IMPLEMENTACIÓN DEL ANÁLISIS MODAL ESPECTRAL PARA UNA PLATAFORMA DE ELEMENTOS FINITOS.

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL

JOSÉ TOMÁS HERNÁNDEZ EYSSAUTIER

PROFESOR GUÍA:

FABIÁN ROJAS BARRALES

MIEMBROS DE LA COMISIÓN:

JUAN FELIPE BELTRAN MORALES

LEONARDO MASSONE SÁNCHEZ

SANTIAGO DE CHILE

2017

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
POR: José Tomás Hernández Eyssautier.
FECHA: 31 / 05 / 2017
PROF. GUÍA: Fabián Rojas.

IMPLEMENTACIÓN DEL ANÁLISIS MODAL ESPECTRAL PARA UNA PLATAFORMA DE ELEMENTOS FINITOS.

La ingeniería civil en Chile está constantemente generando avances en diseño y modelación de estructuras. En este sentido, los softwares de análisis estructural son herramientas muy importante y ampliamente utilizadas en el rubro. El objetivo de este trabajo es dejar una contribución a una de estas plataformas, SAFE_TB. Ésta fue creada por Rojas en 2012 y ha sido utilizada en docencia e investigación hasta la fecha.

El objetivo de este trabajo es complementar la plataforma SAFE_TB implementando una nueva opción de análisis, el análisis modal espectral. Este es una metodología simplificada de análisis dinámico muy utilizada en diseño. Muchas normas alrededor del mundo recomiendan su uso, incluida la norma de diseño sísmico de edificios en Chile, NCh 433.

La plataforma SAFE_TB, se encuentra desarrollada utilizando la metodología de programación orientada a objetos, por lo tanto, todas las clases incorporadas para el desarrollo del análisis deben ser compatibles con las clases originales. También la organización de las funciones del análisis modal espectral, siguen la estructura de los demás tipos de análisis ya incorporados a la plataforma.

Paralelo al desarrollo del análisis, se desea incorporar a la plataforma, la opción de incluir restricciones de puntos múltiples del tipo diafragma rígido. Estas, son ampliamente utilizadas ya que asimilan el comportamiento de las losas de hormigón armado en edificios.

Agradecimientos:

Le agradezco a mis padres, todo el apoyo y compañía durante estos años. Pese a la distancia, jamás me sentí solo y aun en los momentos más duros estuvieron siempre a mi lado. Sé que no entenderán una sola palabra de lo que se explica en estas páginas, pero ustedes me enseñaron lo fundamental para llevar a cabo este trabajo y es que jamás hay que darse por vencido. Esto lo escuche un millón de veces, pero verlos a ustedes lo guardo en mi mente para siempre.

A mis hermanos por todo el cariño, los chistes y las comidas. A estas alturas del camino universitario solo les puedo decir que no se rindan y que disfruten cada día.

También le agradezco a mis amigos, quienes transformaron mi paso por la universidad en una etapa para recordar de por vida. Ustedes me dieron la energía para venir día a día, ya que transformaban cada mañana, cada tarde y cada noche de estudio en los momentos que hoy recuerdo con tanta nostalgia.

A mis profesores, quienes me entregaron los conocimientos que hoy hacen posible ser ingeniero. Gracias por la paciencia y por esas semanas de pruebas, que cuando acababan se transformaban en una celebración sin fin.

Gracias a mi profesor guía Fabián Rojas por su paciencia y buena disposición. Espero estar a la altura de sus expectativas y que mi contribución a su trabajo, sea de utilidad para los que continúan en SAFE_TB.

Finalmente le agradezco a Natalia Ramírez, quien transformo mis últimos años en la facultad en una alegría infinita. Gracias por la compañía y el inmenso cariño que me entregas a diario.

Tabla de contenido

INTRODUCCIÓN 1

1.1. Motivación 1

1.2. Objetivos 1

1.3. Organización de la memoria 2

REVISIÓN BIBLIOGRÁFICA 4

2.1. Introducción 4

2.2. El método modal espectral 4

2.2.1. Dinámica de estructuras 5

2.2.2. Espectro y pseudo espectro de respuesta 10

2.2.3. Análisis modal espectral 13

2.3. Diafragmas rígidos 14

2.3.1. Diafragmas rígidos en estructuras 3D 16

2.3.1. Matriz de transformación para nodos esclavos 17

2.4. Programación en base a objetos 18

2.5. SAFE_TB 19

2.5.1. Estructura 20

2.5.2. ‘Análisis’ en SAFE_TB 21

IMPLEMENTACIÓN DEL ANÁLISIS MODAL ESPECTRAL 22

3.1. Introducción 22

3.2. Estructura del análisis modal espectral 23

3.2.1. ‘EigenAnalysis’ 23

3.2.2. ‘model’ 24

3.2.3. ‘constraintHandler’ 25

3.2.4 ‘dofNumberer’ 26

3.2.5. ‘analysisModel’ 26

3.2.9. ‘systemOfEquation’ 27

3.3. Función ‘analyze’ 28

3.3.1. Iniciar el análisis 28

3.3.2. Determinar valores iniciales 29

3.3.3. Problema de valores y vectores propios 34

3.3.4. Análisis modal espectral 37

3.3.5. Determinar fuerzas resistentes 42

3.3.6. Determinar desplazamientos 44

IMPLEMENTACIÓN DE LA RESTRICCIÓN DE MÚLTIPLES NODOS TIPO DIAGRAGMA RÍGIDO 46

4.1. Introducción 46

4.2. Estructura de la clase ‘ridigDiaphragmConstraint’ 46

4.3. Funcionamiento de la clase ‘ridigDiaphragmConstraint’ 47

PRESENTACIÓN Y EVALUACIÓN DE RESULTADOS 51

5.1. Introducción 51

5.2. Estructuras en 2D 51

5.2.1. Marco 2D de un piso 51

5.2.2. Marco 2D de un piso con masa distribuida 55

5.2.3. Marco 2D de tres pisos con masa distribuida 59

5.3. Estructuras en 3D 66

5.3.1. Marco 3D de un piso 66

5.3.3. Marco 3D en dos direcciones de un piso con diafragma rígido 72
CONCLUSIONES 79
6.1. Conclusiones 79
BIBLIOGRAFÍA 81
ANEXOS 82
Anexo 1 82
Anexo 2 84
Anexo 3 87
Anexo 4 90
Anexo 5 94

Índice de tablas

Tabla 1, Periodos modelo 1. 53
Tabla 2, Modos de vibrar normalizadas modelo 1. 53
Tabla 3, desplazamientos análisis modal espectral modelo 1. 54
Tabla 4, fuerzas resistentes análisis modal espectral modelo 1. 55
Tabla 5, periodos modelo 2. 56
Tabla 6, modos de vibrar normalizados modelo 2. 57
Tabla 7, desplazamientos análisis modal espectral modelo 2. 58
Tabla 8, fuerzas resistentes análisis modal espectral. 59
Tabla 9, Periodos modelo 3. 61
Tabla 10, modos de vibrar normalizados modelo 3. 62

Tabla 11, desplazamientos análisis modal espectral modelo 3. 64

Tabla 12, fuerzas resistentes análisis modal espectral. 66

Tabla 13, periodos modelo 4. 68

Tabla 14, modos de vibrar normalizados modelo 4. 69

Tabla 15, desplazamientos análisis modal espectral modelo 4. 70

Tabla 16, fuerzas resistentes modelo 4. 72

Tabla 17, Periodos modelo 5. 74

Tabla 18, modos de vibrar normalizados modelo 5. 75

Tabla 19, desplazamientos análisis modal espectral modelo 5. 76

Tabla 20, fuerzas resistentes análisis modal espectral modelo 5. 78

Índice de figuras

Figura 1, Sistema de N-GDL. 6

Figura 2, Acelerograma. 11

Figura 3, espectro de respuesta. 12

Figura 4, losa con carga ortogonal 15

Figura 5, losa con carga transversal 15

Figura 6, diafragma rígido. 16

Figura 7, sistemas de coordenadas 2D y 3D. 33

Figura 8, Modelo 3 marco 2D con masa puntual. 52

Figura 9, Modelo 2 marco 2D. 56

Figura 10, Modelo 4 marco 2D 4 niveles. 60

Figura 11, modelo 4, marco 3D. 67

Figura 12, marco en 3D con diafragma rígido. 73

Índice de ecuaciones

Ecuación 1, Definición de vector de movimiento y matrices de masa, amortiguamiento y rigidez. 6

Ecuación 2, Ecuación de equilibrio de fuerzas para un sistema de N-GDL. 7

Ecuación 3, Propiedad de ortogonalidad entre modos. 7

Ecuación 4, Respuesta del sistema expresada en función de los modos de vibrar 8

Ecuación 5, Desarrollo de la ecuación de equilibrio. 8

Ecuación 6, Definición de matrices generalizadas de masa, amortiguamiento y rigidez.. 8

Ecuación 7, Ecuación de equilibrio de los n sistemas de un GDL. 8

Ecuación 8, Ecuación matricial de equilibrio de fuerza. 9

Ecuación 9, solución por partes del grado de libertad 'r'. 9

Ecuación 10, Factor de participación modal. 9

Ecuación 11, Equilibrio de fuerzas en sistema con excitación en la base. 10

Ecuación 12, Factor de participación modal para un sistema excitado en la base. 10

Ecuación 13, Definición del espectro de respuesta 11

Ecuación 14, valor aproximado de velocidad y aceleración. 12

Ecuación 15, Pseudo espectro de respuesta. 12

Ecuación 16, Respuesta máxima del ‘r’ grado de libertad del sistema. 13

Ecuación 17, Combinación ‘Raíz cuadrada de la suma de los cuadrados’. 13

Ecuación 18, Combinación ‘combinación cuadrática completa’ o ‘CQC’. 14

Ecuación 19, posición de un punto dentro de un diafragma rígido. 17

Ecuación 20, movimiento nodo esclavo con respecto al nodo maestro, diafragma rígido.17

Ecuación 21, movimiento nodo esclavo con respecto al nodo maestro, diafragma rígido.17

Ecuación 22, relación de movimiento nodo esclavo / nodo maestro. 18

Ecuación 23, fuerza estática en nodo esclavo..... 18

Ecuación 24, fuerza estática nodo maestro..... 18

Ecuación 25, determinar valores pasivos. 28

Ecuación 26, matriz de rigidez elemento ‘uniaxial spring’. 31

Ecuación 27, matriz de rigidez elemento 'elastic beam column 2D'. 32

Ecuación 28, matriz local de masa para masa puntual. 33

Ecuación 29, matrices de masa y rigidez y vector de influencia ordenados. 36

Ecuación 30, ecuación de Guyan..... 37

Ecuación 31, función 'eig' MATLAB. 37

Ecuación 32, factor de participación modal. 39

Ecuación 33, desplazamientos y aceleraciones modales del sistema. 39

Ecuación 34, matriz de aceleración modal. 39

Ecuación 35, aceleración y desplazamiento pasivos. 40

Ecuación 36, matriz modal amplificada. 41

Ecuación 37, determinar sub matriz de desplazamientos modales..... 41

Ecuación 38, grados de libertad sometidos a la restricción de diafragma rígido. 47

Ecuación 39, 'ID' en diafragmas rígidos luego de ‘constraintHandler.handle’. 49

Ecuación 40, 'ID' en diafragmas rígidos luego de ‘DOFNumberer.numberDOF’. 49

Ecuación 41, 'ID' en diafragmas rígidos luego de ‘constraintHandler.doneNumberingDOF’. 50

Ecuación 42, efecto de la matriz de transformación..... 50

Índice de diagramas

Diagrama 1, Componentes de SAFE_TB..... 20

Diagrama 2, diagrama de flujo análisis modal espectral. 22

Diagrama 3, clases agregadas para el análisis modal espectral. 23

Diagrama 4, diagrama de flujo 'iniciar análisis' 29

Diagrama 5, diagrama de flujo 'determinar valores iniciales' 30

Diagrama 6, diagrama de flujo 'análisis modal espectral' 38

Diagrama 7, diagrama de flujo 'determinar esfuerzos internos'. 43

Diagrama 8, diagrama de flujo 'determinar desplazamientos'. 45

CAPÍTULO I

INTRODUCCIÓN

1.1. Motivación

Debido a su ubicación geográfica, Chile es uno de los países que presenta mayor actividad sísmica en el mundo y cuenta con dos de los terremotos más grandes registrados en la historia de la humanidad. Pese a esto, es uno de los líderes en construcción de edificios de altura en Latinoamérica, y cuenta con el edificio más alto de la región.

En este sentido, es común que las cargas horizontales debido a sismos controlen el diseño de edificios, y es por esto que el análisis sísmico es un área de gran importancia dentro de la ingeniería estructural. El análisis sísmico, tiene como objetivo, el calcular y analizar el comportamiento de estructuras sometidas a cargas cíclicas.

En Chile, el diseño sísmico de edificios es regulado por la norma chilena NCh 433, que indica los requisitos mínimos para el diseño de estructuras. Dentro de este ámbito, una herramienta fundamental para la ingeniería sísmica son los softwares de análisis estructural; que simplifican el trabajo y permiten un mayor detalle del análisis.

Por esta razón, el objetivo de este trabajo de título es complementar el trabajo realizado en la creación de la plataforma SAFE_TB por Rojas en 2012. Esta plataforma permite el análisis estructural mediante elementos finitos o análisis estructural y a diferencia de otros programas, permite una sencilla incorporación de nuevas clases de elementos, materiales, secciones y métodos de integración, entre otros; de esta manera se logra modelar mejor y más rápidamente distintos tipos de estructuras.

1.2. Objetivos

A través de este trabajo de título, se espera implementar de forma correcta el análisis modal espectral, basado en la NCh433 Of2010 en la plataforma SAFE_TB. Para esto es necesario definir nuevas clases, propiedades y métodos, de tal manera que cumplan el objetivo y sean compatibles con lo ya existentes.

Como objetivo secundario, se desea incorporar a la plataforma la opción de definir diafragmas rígidos. Este tipo de restricción es muy utilizada en el modelamiento de estructuras ya que permite asignar a los elementos de losa un comportamiento cercano a la realidad y reducir el número de grados de libertad de la estructura.

1.3. Organización de la memoria

A continuación, presenta la estructuración de este trabajo, enumerando los capítulos que conforman el informe y resumiendo los contenidos abarcados en cada uno de ellos.

CAPÍTULO I: INTRODUCCIÓN

Este capítulo introductorio, contiene un pequeño resumen de lo que tratará este trabajo de título, junto a esto se presenta la motivación del alumno por el desarrollo de este tema, los objetivos a lograr, también la estructuración y contenido de los capítulos que forman este informe.

CAPÍTULO II: REVISIÓN BIBLIOGRÁFICA

El análisis bibliográfico está compuesto por cuatro tópicos principales, que son la investigación sobre el análisis modal espectral, la teoría sobre los diafragmas rígidos en el análisis matricial, la programación en base a objetos y el funcionamiento de la plataforma SAFE_TB. A su vez, este capítulo, se compone por sub temas que serán necesarios para abordar de manera completa los objetivos principales y secundarios que se han propuesto.

CAPÍTULO III: IMPLEMENTACIÓN ANÁLISIS MODAL ESPECTRAL

La plataforma SAFE_TB, está compuesta por diferentes clases implementadas en el software MATLAB, las que se encuentran ordenadas según su propósito. La implementación del método de análisis que se presenta en este trabajo de título seguirá el mismo orden. Este capítulo contiene una explicación de cada una de las clases que componen el análisis y también como trabajan de manera global en la plataforma.

CAPÍTULO IV: IMPLEMENTACIÓN DE LA RESTRICCIÓN DE MULTIPLES NODOS TIPO DIAGRAFMA RÍGIDO

La clase que genera esta restricción se llama ‘rigidDiaphragmConstraint’ y pertenece al grupo de los ‘MP_Constraint’ dentro de ‘model’. En este capítulo se explica cómo funciona internamente esta clase y todas aquellas que se encuentren relacionadas. También como se debe utilizar este tipo de restricción, como se define, cuál es su información de entrada y cómo influye sobre el análisis modal espectral.

CAPÍTULO V: PRESENTACIÓN Y EVALUACIÓN DE RESULTADOS

En este capítulo se exponen y comparan los resultados obtenidos de una serie de estructuras, compuestas por distintos elementos, modeladas en SAFE_TB y SAP 2000. De esta manera se verifica que los resultados obtenidos se encuentran en un rango de error aceptable y el método de análisis modal espectral puede ser utilizado para cualquier sistema.

CAPÍTULO VI: CONCLUSIONES

Por último, en este capítulo de cierre, se presentan las conclusiones, recomendaciones y comentarios sobre este trabajo de título.

CAPÍTULO II

REVISIÓN BIBLIOGRÁFICA

2.1. Introducción

En el siguiente capítulo se muestra una revisión sobre los temas que se tratan en este trabajo de título. Estos, se encuentran separados en 4 partes que son; análisis modal espectral, diafragmas rígidos, programación en base a objetos y SAFE_TB.

En la primera, se muestra la teoría sobre dinámica de estructuras involucrada en este tipo de análisis. También se introduce el concepto de espectro y pseudo espectro de respuesta y una última parte que corresponde al análisis modal espectral propiamente tal.

Luego la parte de diafragmas rígidos, explica cómo estos trabajan en un espacio 3D y como se definen sus variables cuando se utiliza el análisis matricial. Más adelante, en el capítulo de implementación de estos, se muestra como trabajan en SAFE_TB.

Finalmente, el capítulo de programación en base a objetos, entrega una idea global de cómo funciona este método. De esta manera, se comprende mejor la última parte, en donde se explica SAFE_TB; que se encuentra programada de esta manera.

2.2. El método modal espectral

El análisis modal espectral es un método simplificado para el análisis dinámico de estructuras, el cual en lugar de obtener la respuesta en el tiempo; calcula la respuesta máxima de aceleración y desplazamiento. Esto permite obtener los esfuerzos máximos a los que estará sometida la estructura producto de cargas sísmicas de una manera más sencilla. La respuesta máxima del sistema se obtiene como la combinación de las respuestas máximas de cada uno de los modos más significativos de la estructura. Pese a su sencillez, este método proporciona buenos resultados y es recomendado por la mayoría de las normas sísmicas.

El método modal espectral se basa en la premisa que, en análisis lineal, cualquier sistema de N-GDL se puede expresar como la superposición de *n* sistemas de un grado de libertad. Luego, mediante el uso del espectro sísmico de respuesta, es posible determinar la respuesta máxima de cada uno de estos sistemas y así la respuesta del sistema global.

Para entender mejor esta idea, se hará una revisión más detallada del análisis dinámico de estructuras.

2.2.1. Dinámica de estructuras

El método modal espectral surge a partir de la solución de un sistema de n grados de libertad sometido a una excitación arbitraria en el tiempo, en la base. A su vez, este sistema es descompuesto por n sistemas de un grado de libertad. Para poder comprender de forma correcta el análisis y de donde provienen los resultados que este entrega, se estudiará el análisis dinámico de un sistema de N-GDL.

2.2.1.1. Sistemas N-GDL

Un sistema de N-GDL es aquel donde para describir su movimiento es necesaria más de una variable, y en donde el movimiento de cada variable puede afectar el de las demás. Este movimiento puede ser de rotación o desplazamiento, y a su vez puede ser en cualquier dirección espacial. También, un sistema de N-GDL, puede ser continuo o discreto; este dependerá del movimiento y la forma que tenga el sistema a estudiar.

En este sentido; es común modelar sistemas continuos como sistemas discretos, de esta manera se simplifica el análisis obteniendo resultados aceptables. Por ejemplo, el caso de un edificio, se puede modelar como una estructura discreta, asumiendo el desplazamiento horizontal de cada losa como un grado de libertad, como se muestra en la Figura 1. Esto es posible ya que estos son los GDL que mueven mayor cantidad de masa, y el movimiento de otros elementos no contribuye significativamente al movimiento de la estructura completa. Por otro lado, el comportamiento de las losas, se debe a la gran rigidez que estas presentan en su propio plano. Esto está relacionado con el concepto de diafragma rígido, tema que será abordado más adelante.

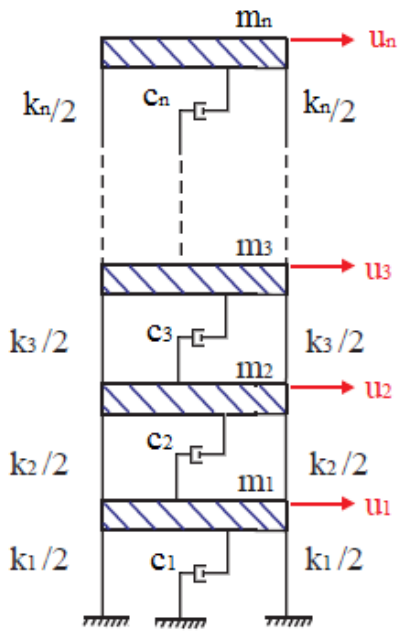

Figura 1, Sistema de N-GDL.[1]

Al igual que para el sistema de un GDL, podemos determinar la ecuación de movimiento expresando el equilibrio dinámico, solo que en este caso la solución es un vector que contiene el movimiento de cada grado de libertad y la masa, amortiguamiento y rigidez son matrices que indican las propiedades del sistema completo, como se muestra a continuación.

$$\{u(t)\} = \begin{Bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ \vdots \\ u_n(t) \end{Bmatrix}, \qquad [m] = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{bmatrix},$$

$$[c] = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c & \cdots & c_{nn} \end{bmatrix}, \qquad [k] = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ k_{n1} & k_{n2} & \cdots & k_{nn} \end{bmatrix}$$

Ecuación 1, Definición de vector de movimiento y matrices de masa, amortiguamiento y rigidez.

[1] Rojas, F. (2015) CI4203: Dinámica de estructuras. Santiago

En las matrices *m*, *c* y *k*, la componente $X_{i,j}$, representa la fuerza inercial, de amortiguamiento o elástica; en el grado de libertad *i*, producto de una aceleración, velocidad o desplazamiento unitario respectivamente, en el grado de libertad *j*.

Una vez determinadas estas matrices, se puede expresar la ecuación de equilibrio de fuerzas de la siguiente manera:

$$[m]\{\ddot{u}(t)\} + [c]\{\dot{u}(t)\} + [k]\{u(t)\} = \{F(t)\}$$

Ecuación 2, Ecuación de equilibrio de fuerzas para un sistema de N-GDL.

A partir de esta ecuación y al igual que para un sistema de un GDL, se determinarán las frecuencias naturales, pero al tratarse de *n* grados, se encuentran *n* frecuencias naturales y modos de vibrar.

Mediante la resolución del problema $\left[[k] - \{\omega\}^2[m]\right][\varphi] = 0$, que corresponde al problema de valores y vectores propios, se obtienen las frecuencias naturales $\{\omega\}_i$ y los modos de vibra del sistema $[\varphi]_i$.

Luego, se define la matriz modal $[\varphi]$, como la matriz que contiene los *n* vectores propios. Esta cumple la propiedad de ortogonalidad, esto quiere decir:

$$[\varphi]^T{}_i[X][\varphi]_j = 0 \; \forall \; i \neq j$$

Ecuación 3, Propiedad de ortogonalidad entre modos.

Esta propiedad es fundamental para la resolución de un sistema de N-GDL ya que permite desacoplar el sistema, lo que se traduce en la diagonalización de las matrices de masa, amortiguación y rigidez.

De esta manera, se puede expresar la solución de la estructura de una nueva manera. Si suponemos que existe una solución para cada sistema independiente $q_i(t)$, que al ser multiplicada por su modo de vibrar correspondiente $[\varphi]_i$, se obtiene la manera en que ésta afecta a los N-GDL de la estructura. Al sumar el efecto de cada una de estas, se obtiene la solución para todo el sistema. Esto se puede expresar de la siguiente manera:

$$\boldsymbol{u(t)} = \sum \{\boldsymbol{\varphi}\}_i \boldsymbol{q_i(t)} = [\boldsymbol{\varphi}]\{\boldsymbol{q(t)}\}$$

Ecuación 4, Respuesta del sistema expresada en función de los modos de vibrar

Luego reemplazando esta solución en la ecuación de equilibrio de fuerzas y pre multiplicando por $[\varphi]^T$ se obtiene:

$$[m]\,[\varphi]\{\ddot{q}(t)\} + [c][\varphi]\{\dot{q}(t)\} + [k][\varphi]\{q(t)\} = \{F(t)\}$$

$$\underbrace{[\varphi]^T[m]\,[\varphi]}_{[M_m]}\{\ddot{q}(t)\} + \underbrace{[\varphi]^T[c][\varphi]}_{[C_m]}\{\dot{q}(t)\} + \underbrace{[\varphi]^T[k][\varphi]}_{[K_m]}\{q(t)\} = [\varphi]^T\{F(t)\}$$

Ecuación 5, Desarrollo de la ecuación de equilibrio.

A partir de esta ecuación, se definen las matrices generalizadas de masa, amortiguamiento y rigidez de la siguiente manera.

$$[M_m] = [\varphi]^T[m]\,[\varphi]$$
$$[C_m] = [\varphi]^T[c]\,[\varphi]$$
$$[K_m] = [\varphi]^T[k]\,[\varphi]$$

Ecuación 6, Definición de matrices generalizadas de masa, amortiguamiento y rigidez.

Como se mencionó anteriormente, la matriz modal cumple la propiedad de ortogonalidad, entonces las matrices generalizadas son matrices diagonales. De esta manera se puede desvincular el sistema de N-GDL y trabajarlo como *n* sistemas de un GDL de la siguiente manera.

$$M_{m_{11}}\ddot{q_1}(t) + C_{m_{11}}\dot{q_1}(t) + K_{m_{11}}q_1(t) = [\varphi]_1{}^T\{F(t)\}$$
$$\dots$$
$$M_{m_{nn}}\ddot{q_n}(t) + C_{m_{nn}}\dot{q_n}(t) + K_{m_{nn}}q_n(t) = [\varphi]_n{}^T\{F(t)\}$$

Ecuación 7, Ecuación de equilibrio de los n sistemas de un GDL.

Este mismo sistema puede expresarse matricialmente de la siguiente manera:

$$[M_m]\{\ddot{q}(t)\} + [C_m]\{\dot{q}(t)\} + [K_m]\{q(t)\} = \{P(t)\}$$

Con $\{P(t)\} = [\varphi]^T\{F(t)\}$.

Ecuación 8, Ecuación matricial de equilibrio de fuerza.

Cada una de estas ecuaciones de movimiento puede ser resuelta de manera independiente; para luego formar el vector *q(t)*, que incluye las soluciones de cada sistema. Con esto, se define la solución global del sistema, como se indica en la Ecuación 4.

Por otro lado, la respuesta particular de $\{q\ (t)\}$ puede ser descompuesta en dos partes. De esta manera, se supone que existen *n* funciones $R_r\ (t)$, las cuales nos indican como varían en el tiempo las *n* componentes de $\{q\ (t)\}$. Y también, *n* constantes Γ_r; que nos indican la amplitud de cada una de las respuestas. Este último, se define como el factor de participación modal, que corresponde al desplazamiento estático y se muestra en la Ecuación 10.

$$q_r(t) = \Gamma_r * R_r(t)$$

Ecuación 9, solución por partes del grado de libertad 'r'.

$$\boldsymbol{\Gamma_r = \frac{P_0}{K_r} = \frac{\{\varphi\}^T{}_r\{F_0\}}{\{\varphi\}^T{}_r[k]\{\varphi\}_r}}$$

Ecuación 10, Factor de participación modal.

El caso en particular que nos interesa para el análisis modal espectral, es el de un sistema de ‘n’ grados de libertad sometido a una aceleración en la base. En este se define un sistema de coordenadas absolutas y otro de coordenadas relativas. Luego, se definen las fuerzas inerciales, de amortiguamiento y elásticas en función de la aceleración, velocidad o desplazamiento respectivamente. Las fuerzas inerciales son definidas con respecto a las coordenadas absolutas, ya que es la aceleración total la que genera la fuerza sobre la masa, mientras que las fuerzas de amortiguamiento y elásticas son una función de las coordenadas relativas ya que ambas propiedades dependen del movimiento relativo entre los grados de libertad. Esto se expresa en la ecuación de equilibrio de fuerzas de la siguiente manera:

$$[m]\{\ddot{u_a}(t)\} + [c]\{\dot{u_r}(t)\} + [k]\{u_r(t)\} = 0$$

$$[m]\{\ddot{u_r}(t) + \ddot{u_g}(t)\} + [c]\{\dot{u_r}(t)\} + [k]\{u_r(t)\} = 0$$

$$[m]\{\ddot{u_r}(t)\} + [c]\{\dot{u_r}(t)\} + [k]\{u_r(t)\} = -[m]\{\ddot{u_g}(t)\}$$

Donde:

u_a: $desplazamiento\ absoluto$

u_r: $desplazamiento\ relativo$

$\ddot{u_g}$: $aceleración\ del\ suelo$

Ecuación 11, Equilibrio de fuerzas en sistema con excitación en la base.

De esta manera, queda definida la misma ecuación de equilibrio de fuerzas que la de un sistema de N-GDL sometido a una fuerza externar $\{F(t)\} = -[m]\{\ddot{u_g}(t)\}$. Por lo tanto, la solución $u(t) = [\varphi]\{q(t)\}$ sigue siendo válida y el factor de participación modal, en este caso, se determina como se muestra en la Ecuación 12.

$$\Gamma_r = \frac{P_0}{K_r} = \frac{\{\varphi\}^T{}_r[m]\{\psi\}}{\{\varphi\}^T{}_r[k]\{\varphi\}_r}$$

Ecuación 12, Factor de participación modal para un sistema excitado en la base.

Aquí, $\{\psi\}$ representa el vector de influencia. Este valor nos indica cómo afecta la aceleración en la base a cada grado de libertad y se cumple que $\{\ddot{u_g}(t)\} = \{\psi\}\,\ddot{u_g}(t)$.

2.2.2. Espectro y pseudo espectro de respuesta

La actividad sísmica puede ser medida mediante el uso de acelerogramas. Estos representan la aceleración producida por el paso de las ondas sísmicas por un punto en la superficie. En la Figura 2, se muestra un acelerograma típico, en este se puede ver que la aceleración del suelo varía entre positivos y negativos, y su amplitud cambia en el tiempo.

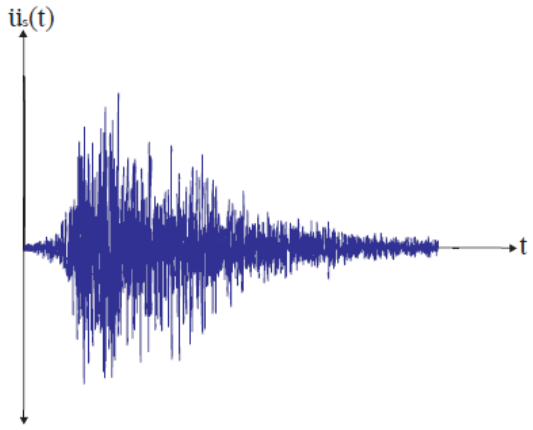

Figura 2, Acelerograma.[2]

Como se ha mencionado anteriormente, a partir de una aceleración en la base, es posible determinar la respuesta en el tiempo para la aceleración, velocidad y desplazamiento. Sin embargo, para el cálculo estructural, que se basa en los esfuerzos máximos a los que se someterá el sistema, no es necesario conocer la respuesta en cada instante de tiempo. Bastará con conocer los máximos de ésta para determinar los esfuerzos máximos.

En 1932, Maurice Biot (1905-1985) desarrollo el concepto de espectro de respuesta. Esto corresponde al registro de las máximas respuesta obtenidas a partir de un acelerograma para cada periodo posible de un oscilador de un grado de libertad. Este puede ser un espectro de aceleración ($S_a\ (T)$), velocidad ($S_v\ (T)$) o desplazamiento ($S_d\ (T)$).

De esta manera se define para un sistema de un grado de libertad y periodo T_i, la respuesta máxima para un registro $u_g(t)$ de la siguiente manera:

$$Max(|u(t)|) = S_d(T_i)$$
$$Max(|\dot{u}(t)|) = S_v(T_i)$$
$$Max(|\ddot{u}(t)|) = S_a(T_i)$$

Ecuación 13, Definición del espectro de respuesta

[2] Rojas, F. (2015) CI4203: Dinámica de estructuras. Santiago

Estos máximos pueden ser graficados en función de los periodos, obteniendo un gráfico del espectro de respuesta. En la Figura 3 se muestra un espectro de aceleraciones, donde se representa la aceleración máxima en función de la frecuencia de distintos sistemas.

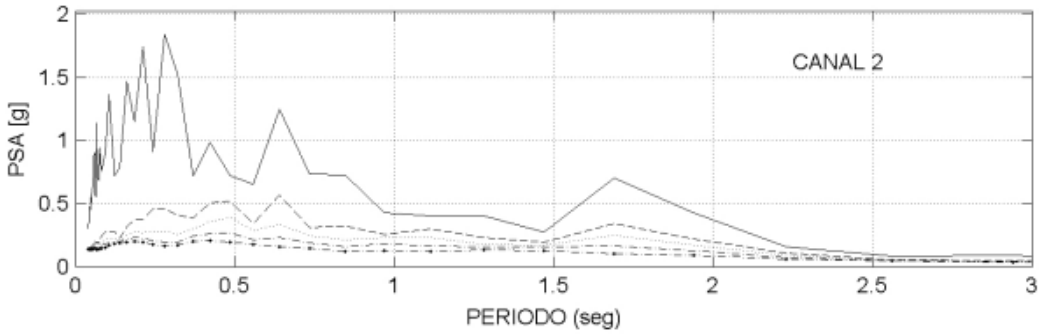

Figura 3, espectro de respuesta.[3]

Típicamente, la razón de amortiguamiento β del sistema, se encuentra del orden del 2 al 10 %, por lo tanto, se puede aproximar $(1-\beta^2) \approx 1$. Utilizando esto, se determina un valor aproximado para la velocidad y la aceleración de la siguiente manera:

$$\dot{u}_r(t) \approx \omega_n\, u(t) \quad y \quad \ddot{u}_r(t) \approx {\omega_n}^2\, u(t)$$

Ecuación 14, valor aproximado de velocidad y aceleración.

Luego, se define el pseudo espectro de respuesta, obteniendo los máximos de estas ecuaciones.

$$PS_v(T) = \omega_n\, S_d(T)$$
$$PS_a(T) = {\omega_n}^2\, S_d(T)$$

$$PS_v(T) = S_a(T)/\omega_n$$
$$PS_d(T) = S_a(T)/{\omega_n}^2$$

Ecuación 15, Pseudo espectro de respuesta.

[3] Boroschek, R; Soto, P; Leon, R (2010): Informe red local de registros de edificios 'Cámara chilena de la construcción'. Santiago, edificio 'Cámara chilena de la construcción', febrero 27, 2010 hora 3:34.

2.2.3. Análisis modal espectral

Como se ha mencionado, el análisis modal espectral es un método simplificado de análisis dinámico de estructuras, utilizado para determinar el máximo valor para la respuesta de sistemas de N-GDL.

En la Ecuación 9, fue definida la solución de la ecuación 'r' de un sistema de N-GDL. Mediante esta ecuación podemos determinar el máximo de la respuesta de la siguiente manera:

$$u_r(t) = [\varphi]_r q_r(t) = [\varphi]_r \Gamma_r R_r(t)$$

$$Max(|q_r(t)|) = Max(|\Gamma_r R_r(t)|) = \Gamma_r Max(|R_r(t)|)$$

Donde,

$$Max(|R_r(t)|) = S_d(T_r)$$
$$Max\left(\left|\ddot{R}_r(t)\right|\right) = S_a(T_r)$$

Por lo tanto,

$$q_{r_{max}}(t) = \Gamma_r S_d(T_r)$$
$$\ddot{q}_{r_{max}}(t) = \Gamma_r S_a(T_r)$$

Ecuación 16, Respuesta máxima del 'r' grado de libertad del sistema.

Una vez determinada la respuesta máxima de cada uno de los grados de libertad que forman el sistema, se podría sumar el efecto de cada una de estas utilizando la formula $u_{max}(t) = [\varphi]\, q_{max}(t)$, pero sería equivalente a suponer que todos los máximos ocurren al mismo tiempo, lo que es incorrecto. Por este motivo se utilizan distintos tipos de combinaciones modales.

Algunas de estas combinaciones son:

$$\{u(t)\} = \sqrt{\sum_{r=1}^{N} \{u_r{}^2(t)\}}$$

Ecuación 17, Combinación 'Raíz cuadrada de la suma de los cuadrados'.

$$\{u(t)\} = \sqrt{\sum_{r=1}^{N}\sum_{s=1}^{N}\{u_r(t)\}\rho_{rs}\{u_s(t)\}}$$

$$donde,$$

$$\rho_{rs} = \frac{8\beta^2\lambda^{3/2}}{(1+\lambda)(1-\lambda)^2 + 4\lambda\beta^2(1+\lambda)}$$

Ecuación 18, Combinación 'combinación cuadrática completa' o 'CQC'.

La combinación CQC es el método de superposición modal utilizado en la norma NCH 433 y el que se implementa durante este trabajo.

2.3. Diafragmas rígidos

El concepto de diafragma rígido, se utiliza típicamente en edificios en donde la existencia de losas caracteriza el movimiento cada nivel. Así, aunque este elemento presente una naturaleza flexible en el sentido ortogonal, exhibe una gran rigidez en su propio plano. Esto se explica mejor en la Figura 4 e Figura 5. En estas se muestra una estructura compuesta por 4 columnas y una losa en la parte superior.

En la primera, la losa se encuentra cargada en sentido vertical. Aquí, la deformación aumenta a medida que se aleja de los apoyos, de esta manera, se puede apreciar que en el centro de la losa la deformación es máxima y que el elemento es flexible. Por otro lado, en la Figura 5, se muestra la misma losa sometida a una carga lateral, como sería el caso de las cargas sísmicas o cargas de viento. Como se puede ver, la deformación, no genera extensión entre puntos de la losa. Esto se debe a la gran rigidez que presenta en este sentido.

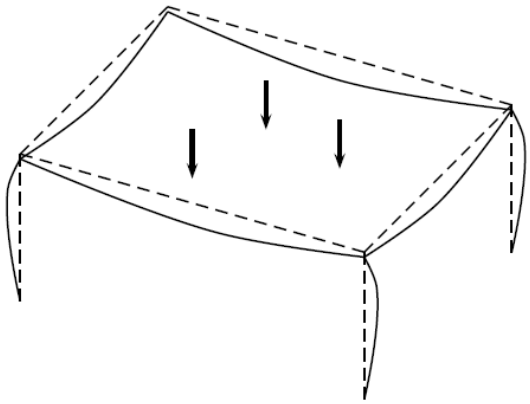

***Figura 4, losa con carga ortogonal*[4]**

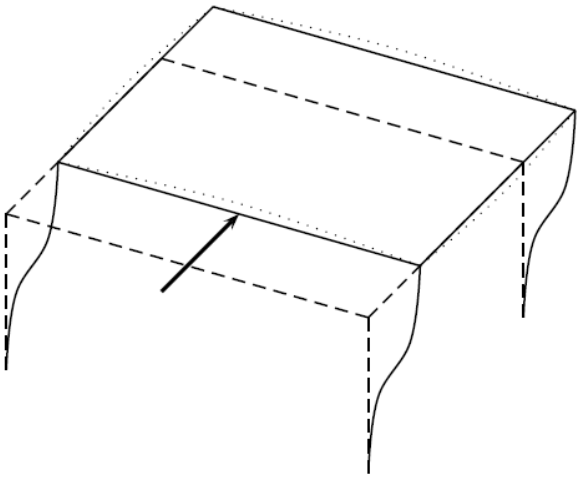

***Figura 5, losa con carga transversal.*[5]**

Como es esperable, existe una relación geométrica entre los desplazamientos horizontales de la losa y a partir de esta se modifican las matrices de masa y rigidez. A continuación, se explica el efecto de los diafragmas rígidos en la estructura.

[4] Hurtado, J. ( ) Análisis matricial de estructuras. Bogotá.

[5] Hurtado, J. ( ) Análisis matricial de estructuras. Bogotá.

2.3.1. Diafragmas rígidos en estructuras 3D

El movimiento en el plano de la losa, puede ser representado con 3 variables asociadas a un nodo principal. Dos corresponden al desplazamiento en sentido horizontal, $\vec{x}$ e $\vec{y}$, y la otra al giro en el eje vertical, $\vec{\theta}_z$. Luego, a partir de estas variables se define el movimiento dentro del plano de cualquier nodo perteneciente a la losa, estos se conocen como 'nodos esclavos'.

En la Figura 6, se puede ver un ejemplo de una losa en la que se considera el efecto de un diafragma rígido. En esta, el nodo C representa el nodo maestro, el A un nodo esclavo y el vector $\vec{R}_A$ la posición relativa del nodo A con respecto al nodo C. También, $\vec{u}_c$ corresponde a la suma vectorial de los desplazamientos horizontales $\vec{x}$ e $\vec{y}$ correspondientes al nodo C y $\vec{\theta}_c$ al giro de este mismo.

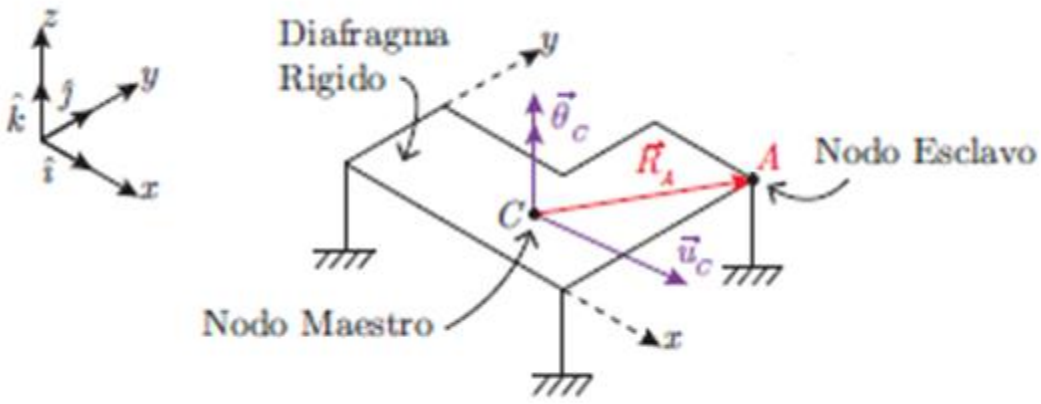

Figura 6, diafragma rígido.[6]

De esta manera, podemos definir para el movimiento dentro del plano horizontal que:

$$\vec{u}_C = u_{Cx}\hat{\imath} + u_{Cy}\hat{\jmath} + u_{Cz}\hat{k}$$

$$\vec{\theta}_c = 0\,\hat{\imath} + 0\,\hat{\jmath} + \theta_{Cz}\hat{k}$$

[6] Rojas, F. ( ) CI5211: Análisis matricial de estructuras. Santiago

$$\vec{R}_A = x_A\hat{\imath} + y_A\hat{\jmath} + 0\,\hat{k}$$

Ecuación 19, posición de un punto dentro de un diafragma rígido.

Y así, determinar el movimiento del nodo esclavo de la siguiente manera.

$$\vec{u}_A = \vec{u}_C + \vec{\theta}_C \times \vec{R}_A$$

Y al resolver el producto cruz de la ecuación obtenemos.

$$\vec{u}_A = \begin{Bmatrix} u_{Ax} \\ u_{Ay} \\ u_{Az} \end{Bmatrix} = \begin{Bmatrix} u_{Cx} + \theta_{Cz} y_A \\ u_{Cy} - \theta_{Cz} x_A \\ u_{AZ} \end{Bmatrix}$$

Ecuación 20, movimiento nodo esclavo con respecto al nodo maestro, diafragma rígido.

Por otro lado, al no existir desplazamiento relativo entre los nodos pertenecientes al diafragma, el giro en el sentido ortogonal es igual en todo el elemento.

$$\vec{u}_A = \begin{Bmatrix} \theta_{Ax} \\ \theta_{Ay} \\ \theta_{Az} \end{Bmatrix} = \begin{Bmatrix} \theta_{Ax} \\ \theta_{Ay} \\ \theta_{Cz} \end{Bmatrix}$$

Ecuación 21, movimiento nodo esclavo con respecto al nodo maestro, diafragma rígido.

Finalmente, para lograr la relación entre los grados de libertad de los nodos maestro y esclavo de define la matriz de transformación de la siguiente manera:

$$[T_{DR}] = \begin{bmatrix} 1 & 0 & Ry \\ 0 & 1 & -Rx \\ 0 & 0 & 1 \end{bmatrix}$$

2.3.1. Matriz de transformación para nodos esclavos

Si bien, ya hemos definido como se obtiene la respuesta para los nodos esclavos a partir del maestro, es también necesario determinar cómo se obtienen los valores iniciales definidos para el nodo maestro a partir de los esclavos; por ejemplo, las matrices de masa y rigidez.

Para esto, partiremos suponiendo que $[K_A]$ es la matriz local de rigidez de un elemento en el cual, uno de sus nodos pertenece a un diafragma rígido. También, basándonos en el desarrollo anterior,

supondremos que los desplazamientos de los nodos esclavos y maestro se encuentran relacionados de la siguiente manera:

$$\{u_A\} = [T_{DR}]\{u_C\}$$

Ecuación 22, relación de movimiento nodo esclavo / nodo maestro.

Por otro lado, sabemos que la fuerza en ese nodo está dada por:

$$\{F_A\} = [K_A]\{u_A\}$$

Ecuación 23, fuerza estática en nodo esclavo.

Y al desarrollarlo, obtenemos que:

$$\{F_A\} = [K_A][T_{DR}]\{u_C\}$$

$$\underbrace{[T_{DR}]^T\{F_A\}}_{\{F_C\}} = \underbrace{[T_{DR}]^T[K_A][T_{DR}]}_{[K_C]}\{u_C\} = \{F_C\}$$

Ecuación 24, fuerza estática nodo maestro.

A partir de esto y debido a la ecuación de restricción, es fácil obtener la matriz de transformación. Solo se debe considerar el número de grados de libertad que tengan los nodos.

2.4. Programación en base a objetos

La programación orientada a objetos, es un método o estilo de programación en el cual se definen estructuras lógicas, las cuales contienen métodos y funciones que definen la naturaleza de los objetos definidos mediante dicha clase. Este enfoque permite una mejor organización y optimización de los elementos programados.

Los objetos son definidos a partir de una determinada clase. En ellas se establece toda la información necesaria para crear el objeto y todas las funciones con las que contará. También, a partir de una clase, se pueden definir cuantos objetos se desee y a cada uno se le puede entregar diferente información de entrada.

Para comprender mejor que es la programación orientada a objetos, primero se definen algunos conceptos.

Clase: Definición de propiedades y métodos de un objeto en particular. Cuenta con un constructor que crea el objeto. Este puede pedir algún tipo de información de entrada al usuario para crear el objeto y a partir de esta formar sus propiedades.

Herencia: El concepto de herencia corresponde a las propiedades y métodos que se transmiten de una clase a otra cuando estas han sido definidas como base, súper clase y sub clase. Una clase se define como hija cuando se trata de una clase similar; tal vez más específica, que se deriva de otra clase. Por lo tanto, las propiedades y funciones de la clase madre se mantienen, salvo que se definan nuevamente en la clase hija, en dicho caso controla lo que esta última diga.

Objeto: Entidad provista de un conjunto de propiedades y métodos definidos en la clase a la que pertenece.

Propiedades: Variables almacenadas en cada objeto que son definidas según la clase a la que pertenecen.

Método: Corresponde a un conjunto de acciones organizadas dentro de una función en una clase. Comúnmente, se utiliza la notación de (.) para ejecutarla, la forma se expresa de la forma 'nombreObjeto.nombreMétodo'. La función puede generar cambios internos en las propiedades del objeto, entregar información al usuario, calcular algún nuevo parámetro, entre otros.

La idea de la programación en base a objetos, es que estos interactúen entre sí. También, se permite que los elementos de entrada o propiedades de algún objeto sean otros objetos. De esta manera, se facilita el traspaso de información ya que si un objeto guarda en sus propiedades otro objeto, automáticamente tendrá acceso a sus propiedades y métodos.

Dicho esto, es más clara la explicación de por qué SAFE_TB se encuentra programado orientado a objetos. En la plataforma, se deben generar los objetos pertenecientes a diferentes clases, como; nodos, materiales, elementos, restricciones, tipos de análisis, entre otros. Estos se relacionan entre sí y permiten formar el sistema estructural que se desea modelar.

2.5. SAFE_TB

SAFE_TB es una plataforma de análisis de elementos finitos implementada en MATLAB que utiliza lenguaje de programación orientada a objetos. Esta plataforma es capaz de modelar

estructuras y simular su comportamiento ante cargas estáticas y dinámicas. También permite la incorporación de materiales con comportamiento no-lineal.

Debido a su estructura y estilo de programación, SAFE_TB resulta sencillo de modificar y ampliar. De esta manera, cualquier usuario puede agregar nuevos elementos, métodos de análisis o lo que fuese necesario.

A continuación, se presenta una revisión general de la organización de la plataforma y cómo trabajan las clases utilizadas en el desarrollo del análisis modal espectral.

2.5.1. Estructura

La plataforma está formada por cuatro componentes principales que son ‘Modelo’, ‘Análisis’, ‘Grabador de Respuesta’ y ‘Manejo Gráfico’. Estas están relacionadas como se indica en el Diagrama 1, en donde la dirección de las flechas indica en qué sentido se produce el intercambio de información.

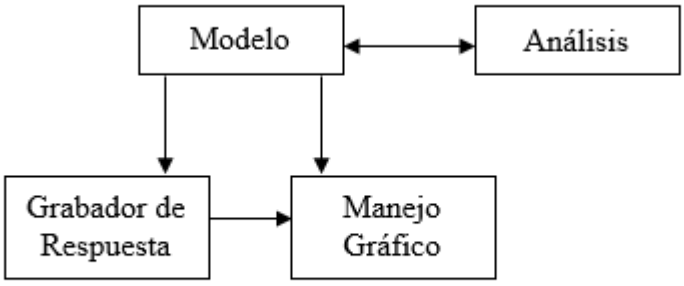

Diagrama 1, Componentes de SAFE_TB

De esta manera, la parte ‘Análisis’ es la encargada de controlar el estado de ‘Modelo’. Esta última contiene toda la información de la estructura en su estado inicial y actual. ‘Grabador de Respuesta’, es quien almacena todos los estados intermedios y ‘Manejo Gráfico’, entrega la información solicitada de cualquier estado del modelo de manera gráfica.

Este trabajo se centra en la implementación de un método de análisis, por lo tanto, la revisión del funcionamiento de SAFE_TB se enfocó en esa área.

2.5.2. ‘Análisis’ en SAFE_TB

Esta sección, contiene todos los tipos de análisis que incluye SAFE_TB, como estático, dinámico y el método que se presenta en este trabajo: el análisis modal espectral.

Si bien, estos utilizan distintas metodologías, todos poseen una estructura similar. Esto permite entender de manera más sencilla el procedimiento que existe detrás del análisis, generar modificaciones y crear nuevos métodos.

Todos los tipos de análisis se componen por una clase principal y clases secundarias, cabe destacar que esto no tiene relación con el concepto de herencia, sino que la clase principal es la articuladora del análisis, almacena las clases secundarias y utiliza sus propiedades y funciones para su desarrollo.

Las clases secundarias son ‘*AnalysisModel’*; que contiene la información del modelo definido, ‘*ConstraintHandler’*; maneja el intercambio de información de las restricciones entre el modelo y ‘*AnalysisModel’*, ‘*ConvergenceTest’*; contiene el test de convergencia utilizado en los análisis iterativos, ‘*DOFNumberer’*; asigna un número de identificación a cada grado de libertad en ‘*AnalysisModel’*, ‘*ElementHandler’*; maneja el intercambio de información de los elementos entre el modelo y ‘*AnalysisModel’*, ‘*Integrator’*; para los análisis iterativos maneja el paso al siguiente estado, *NodeHandler’*; maneja el intercambio de información de los nodos entre el modelo y ‘*AnalysisModel’*, ‘*SolutionAlgorithm’*; maneja la solución de cada estado de la iteración, ‘*SystemOfEquation’*; resuelve el sistema y ‘*AnalysisGraph’*; maneja la representación gráfica de ‘*AnalysisModel’*.

La clase principal, que como se dijo; es la gestora del análisis, vincula y almacena las clases secundarias, de esta manera tiene acceso a sus propiedades y funciones. También, la clase principal contiene la función de ‘*analize’*, en la que se ejecuta el algoritmo que resuelve el análisis. Dentro de esta, se ejecutan todos los objetos secundarios necesarios para desarrollas el objetivo, como por ejemplo determinar los valores iniciales, resolver el sistema y agregar el estado final al modelo.

CAPÍTULO III

IMPLEMENTACIÓN DEL ANÁLISIS MODAL ESPECTRAL

3.1. Introducción

Como se dijo anteriormente, cada análisis tiene una clase principal y varias clases secundarias. En este capítulo se explica cómo funcionan, cómo están relacionadas y que función cumplen dentro del análisis modal espectral.

En el Diagrama 2, se muestra un diagrama de flujo en el que se indican las principales etapas del desarrollo del análisis.

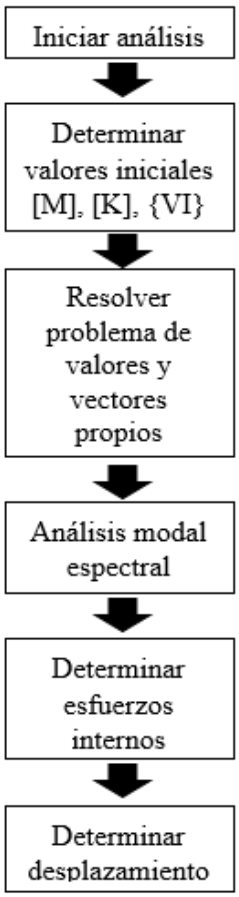

Diagrama 2, diagrama de flujo análisis modal espectral.

Éste comienza con una etapa de inicio, luego se determinan los valores iniciales; los cuales son determinados a partir del modelo definido. Después, se desarrolla el problema de valores y vectores

propios, que como se explicó anteriormente, se utiliza para determinar los periodos de la estructura y las formas modales. A continuación, se resuelve lo que corresponde al análisis modal espectral propiamente tal, en esta etapa se incorpora el espectro de respuesta y se determinan los desplazamientos y aceleraciones por modos. Finalmente, se determinan los desplazamientos y aceleraciones por cada grado de libertad y los esfuerzos de cada elemento perteneciente al sistema usando un método de combinación modal. Estos últimos pasos determinan los resultados que serán entregados a usuario al concluir el método.

En este capítulo se explica cada una de las clases que forman el análisis, como se definen y como trabaja el constructor de cada una de ellas. Luego se habla de la función '*analyze'*, donde se explica el algoritmo que esta comprende y como esta se comunica con los objetos secundarios.

3.2. Estructura del análisis modal espectral

Los objetos que se muestran en este sub capítulo, corresponden a las clases que intervienen en el desarrollo del análisis modal espectral. Algunas de ellas fueron previamente definidas para otros análisis y se reutilizan en este.

En el Diagrama 3, se pueden ver que clases fueron definidas exclusivamente para este trabajo de título y a que parte de la estructura de la plataforma corresponden

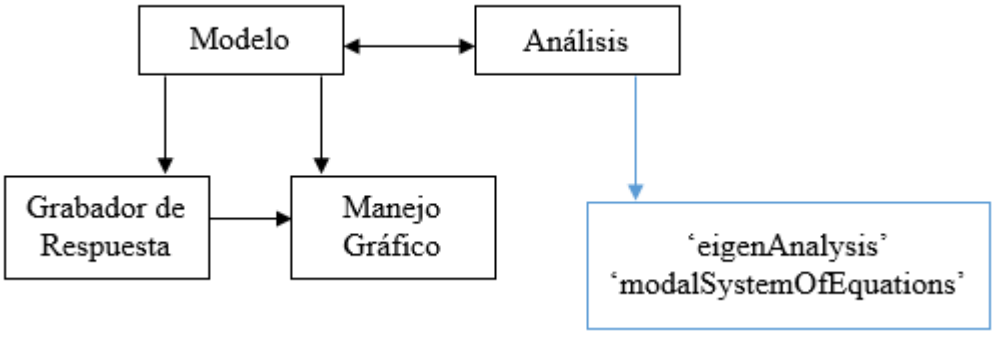

Diagrama 3, clases agregadas para el análisis modal espectral.

3.2.1. '*EigenAnalysis'*

'*EigenAnalysis'* es la clase principal en el método de análisis modal espectral. Para definir un objeto perteneciente a esta clase, primero se deben definir sus objetos de entrada, estos son: '*model'*,

‘*constraintHandler’*, ‘*DOFNumberer’*, ‘*analysisModel’*, ‘*integrator’* y ‘*systemOfEquation’*. También se debe definir el espectro de diseño, como se indica en la norma NCh 433, creando una matriz de dos columnas en donde la primera corresponde al período y la segunda a la aceleración, y se debe indicar la dirección en la que se desea considerar el sismo.

Lo primero que verifica el constructor de ‘*EigenAnalysis’* es que se hayan entregado todos los elementos de entrada. De no ser así, este crea elementos vacíos y el análisis no podrá ser ejecutado.

Luego guarda los elementos en las propiedades del objeto y ejecuta la función ‘*setLink’* de ‘*analysisModel’*, ‘*constraintHandler’* y ‘*DOFNumberer’*. Esta función permite que estos objetos adquieran algunas de sus propiedades a partir de la información que se encuentra guardada en ‘*eigenAnalysis’*.

Aparte del constructor y de ‘*analyze’*, existen otras funciones necesarias para el desarrollo del análisis. Una de estas es ‘*clearAll’*; su propósito es borrar las diferentes componentes del análisis asociado a ‘*eigenAnalysis’*. También ‘*modelChange’*, esta se encarga de incorporar todos los posibles cambios que haya tenido el modelo desde que se definió el modelo en el constructor.

Finalmente, las funciones ‘*get’* y ‘set’ se utilizan para obtener y modificar las propiedades del objeto desde fuera de él. Esto se debe ya que ‘*eigenAnalysis’* es una clase con sus propiedades de acceso privado, esto significa que sus propiedades solo pueden ser obtenidas y modificadas desde dentro de la clase.

3.2.2. ‘*model’*

Como se mencionó, este objeto contiene toda la información correspondiente al modelo definido por el usuario. Es por esto, que cuando se desea agregar un nuevo elemento; se debe crear el objeto correspondiente y luego agregarlo al objeto ‘*model’*

El constructor, no recibe ningún tipo de información de entrada, por lo tanto, crea todas las variables como elementos vacíos. Luego, con las funciones ‘*add’*, se guardan los elementos en las propiedades ‘*storage’*, dependiendo del tipo de elemento que se trate. Para obtener cualquier objeto almacenado en el modelo, se deben utilizar las funciones ‘*get’*.

3.2.3. ‘*constraintHandler*’

Existen dos opciones al momento de definir este objeto, y el que será utilizado depende si el modelo considera diafragmas rígidos u otro tipo de restricción múltiple (‘*MP_Constraint’*). Si no existen ‘MP_C’ se puede utilizar la clase ‘*BasicConstraintHandler’*, de lo contrario se debe utilizar ‘*TransformationConstriaintHandler’*.

Como ya se ha mencionado, este objeto es el encargado de manejar las restricciones impuestas en el modelo, pero también es quien define y guarda los ‘*NodeHandler’* y ‘*ElementHandler’*.

Al definir un objeto de esta clase, no se le entrega ningún tipo de información de entrada, por lo tanto, el constructor solo crea las propiedades del objeto como elementos en blanco. Es por esto, que la función ‘*setLink’*; que ya mencionamos en el constructor de ‘*eigenAnalysis’*, es la encargada de incorporar la información del ‘*model’*, ‘*analysisModel’* e ‘*integrator’* a las propiedades del ‘*ConstraintHandles’*.

Otra función importante dentro de esta clase es ‘*handle’*, aquí se crean los ‘*nodeHandler’*; uno por cada nodo y los ‘*elementHandler’*; uno por cada elemento del sistema. Estos objetos se utilizan para determinar y almacenar la información del elemento que tiene relación con el análisis, como matrices de rigidez y masa, grados de libertad, entre otros.

Una propiedad muy importante dentro del ‘*NodeHandler’* es el ‘*nodeID’*. Esto corresponde a un vector que contiene un número de identificación para cada grado de libertad del nodo. Este número es único en todo el sistema y permite trabajar de manera ordenada durante el análisis. Más adelante explicaremos como se enumera el sistema y que usos tiene este vector. ‘*ConstraintHandler’* es quien maneja el etiquetado de aquellos grados de libertad sometidos a algún tipo de restricción. De esta manera, se etiqueta con un ‘-1’ si se encuentra asociado a un ‘SP_*Constraint’* o un ‘-2’ si es a un ‘*MP_Constraint’*.

Finalmente, y al igual que otras clases, esta cuenta con funciones de ‘*get’* y ‘set’ para modificar y obtener información del objeto.

3.2.4 ‘*dofNumberer’*

Al igual que para la clase ‘*constraintHandler’*, existen dos opciones para definir ‘*dofNumberer’* dependiendo si existen o no restricciones múltiples en el sistema. De existir se debe utilizar la clase ‘*transformationDOFNumberer’*, o de lo contrario se puede usar ‘*BasicDOFNumberer’*.

El constructor de esta clase no recibe ningún tipo de información de entrada, por lo que crea todas sus propiedades como elementos vacíos. Es la función ‘*setLink’* quien maneja la entrada de información al objeto y almacena el ‘*analysisModel’* en las propiedades.

Como dijimos anteriormente, el ‘*nodeID’* es un etiquetado que enumera todos los grados de libertad del sistema y permite trabajar de manera ordenada. El ‘*dofNumberer’* es quien maneja dicho sistema, y es el encargado de asignarle el ‘ID’ a cada grado de libertad.

Para esto se debe ejecutar la función ‘*numberDOF’*, esta primero que todo; llama a ‘*analysisModel’*, para poder tener acceso a ‘*constraintHandler’* y ‘*nodeHandler’*. Luego itera recorriendo todos los nodos y sus grados de libertad, los enumera y se salta aquellos que cuenten con una restricción.

3.2.5. ‘*analysisModel’*

Esta clase es la encargada de controlar la información proveniente del modelo dentro de la etapa de análisis. Esto se logra, principalmente, gracias al manejo de los objetos de tipo ‘*handler’*, como ‘*nodeHandler’*, ‘*contrainitHandler’* y ‘*elementHandler*. Estos son almacenados en el ‘*analysisModel’* y entregan la información correspondiente al elemento que representan.

El constructor de la clase, no recibe ningún tipo de información de entrada. Por lo tanto, crea todas las variables del objeto como elementos vacíos. Luego, la función ‘*setLink’* es la encargada de integrar la información del modelo dentro del objeto. Esta recibe como información de entrada; el modelo definido por el usuario y el ‘*constraintHandler’*.

Las propiedades de esta clase son; ‘*model’*, ‘*constraintHandler*, ‘*numOfEq’*, ‘*numNodeHandler’*, ‘*hasAnalysisModelChanged’*, ‘*nodeHandlerStorage’*, ‘*elementHandlerStorage’*, ‘*dofGraph’*, ‘*optionalCalculateDOFGraph’*, ‘*optionalCreateDOFGraph’*, ‘*eigenVector’* y ‘*eigenValue’*. Cada una de estas representa algún tipo de información con respecto al modelo, necesaria para el análisis.

Luego las funciones con las que cuenta esta clase, son principalmente para obtener, agregar o modificar la información guardada en las propiedades.

3.2.9. '*systemOfEquation'*

'*SystemOfEquation'* es la clase que se encarga de resolver los sistemas de ecuaciones que se forman durante el análisis. También es quien arma, ordena y almacena las variables correspondientes a estos sistemas, es por esto que sus funciones son de las más utilizadas durante en análisis.

En el análisis modal espectral, es necesario definir el sistema de ecuaciones '*ModelSystemOfEquations'*. Esta es una clase hija de '*SystemOfEquiation'*; por lo que cuenta con todas las funciones y propiedades de la clase madre, pero incluye otra especialmente implementadas para la metodología de análisis modal.

Algunas de sus propiedades son, la matriz de masa y rigidez globales del sistema, estas se encuentran ordenadas según la enumeración del '*nodeID'* e incluyen todos los grados de libertad independientes que no se encuentren restringidos. El vector de influencia, el cual indica cómo afecta el sismo a cada grado de libertad. También, la matriz de masa, rigidez y vector de influencia condensados, los que se utilizan cuando existen grados de libertad pasivos. Las matrices de rigidez pasiva y activa/pasiva, que se utilizan para determinar los valores pasivos como se muestra en la Ecuación 25. '*EigenValue'* y '*EigenVector'*, estos corresponden a los valores y vectores propios respectivamente, que a su vez en el análisis modal; corresponden a la frecuencia angular y las formas de vibrar por modo. La matriz 'Cid', que ordena las matrices de masa y rigidez para que puedan ser condensadas. Es importante guardar esta matriz ya que desde la condensación en adelante el orden de los grados de libertad cambia. '*NumberOfEq'*, guarda el número de grados de libertad del sistema. Finalmente '*dispByMode'* y '*accelByMode'* guardan los desplazamientos y aceleraciones correspondientes al análisis modal espectral antes de hacer la combinación CQC, esto permite determinar otros parámetros a partir de estos valores; ya que esta combinación no se puede realizar suponiendo superposición.

$$y_{global} = \begin{Bmatrix} y_{activos} \\ y_{pasivos} \end{Bmatrix}$$

$$K_{global} = \begin{bmatrix} K_{aa} & K_{ap} \\ K_{pa} & K_{pp} \end{bmatrix}$$

$$y_{pasivos} = {K_{pp}}^{-1} * K_{ap} * y_{activos}$$

Ecuación 25, determinar valores pasivos.

El constructor de esta clase no recibe ningún parámetro de entrada, por lo que solo crea las propiedades como elementos en blanco. Para agregar esta información al objeto, se utilizan las funciones de tipo '*add'*; como '*addTangent'*, '*addMass'*, '*adInfluence'*, entre otras. Estas funciones reciben como información de entrada una matriz local y el '*ID'* del grado de libertad, en base al 'ID' agrega la matriz local en la matriz global. Estas funciones son utilizadas, por ejemplo, en una iteración dentro de la función '*analize'*, entonces al recorrer todos los nodos y elementos; el resultado es la matriz global completa.

3.3. Función '*analyze'*

Como dijimos anteriormente, la función '*analyze'* es la gestora principal del análisis modal espectral. Esto quiere decir, que dentro de ella se ejecutan todos los procesos necesarios para llevar a cabo análisis. A continuación, se explicará cada una de las etapas indicadas en el Diagrama 2 de manera más detallada, indicando sus procesos y que otras clases y funciones se ven relacionadas.

3.3.1. Iniciar el análisis

Esta primera etapa, tiene como objetivo actualizar los cambios que pudiesen existir dentro del modelo; entre que se crea el objeto '*eigenAnalysis'* y se ejecuta la función '*analyze'*. Para esto, se realizan tres procesos que son, obtener el modelo, revisar cambios y actualizar el modelo.

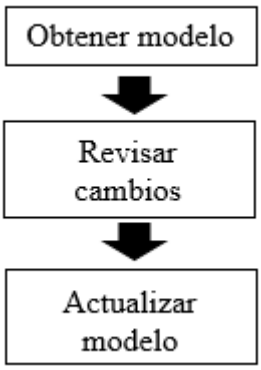

Diagrama 4, diagrama de flujo 'iniciar análisis'

De esta manera, para obtener el modelo, se ejecuta la función '*eigenAnalysis.getModel'*. Esta entrega directamente el modelo que ha sido guardado dentro de las propiedades del '*eigenAnalysis'*.

Una vez adquirida la información del modelo, se procede a revisar si existen cambios en este, para ello se utiliza la función '*model.hasModelChanged'*. Esta entrega la información almacenada en la propiedad '*hasModelPropiertesChanged'* del modelo.

Dentro de la clase '*model'*, existen funciones para agregar o retirar los distintos elementos que lo componen. Estas también modifican la propiedad '*hasModelPropiertesChanged'*, guardando 'true' en ella cada vez que se edita algo. Por lo tanto, esta propiedad es la que indica si existen cambios.

Finalmente, dependiendo de la información obtenida, se procede a actualizar los cambios dentro del objeto de análisis. Para lo cual, se utiliza la función '*eigenAnalysis.modelChanged'*, en esta se procede a reestructurar las propiedades del análisis como '*analysisModel'*, '*constraintHandler'*, '*DOFNumberer'* y '*systemOfEquations'*.

Para esto, cada uno de estos objetos ejecuta su función principal, ajustando todas las propiedades nuevamente e incluyendo las que han sido incorporadas de manera posterior. De esta manera queda incorporada toda la información del modelo y no se producirán incongruencias durante el análisis.

3.3.2. Determinar valores iniciales

En el Diagrama 5, se muestra el proceso para determinar los valores iniciales del sistema. Si bien, éste es como se indica en la figura, también se podría considerar como 3 etapas independientes, las

que corresponden a formar la matriz de masa, de rigidez y el vector de influencia. Como el proceso para formar los valores locales son diferentes, se explicará cada una por separado, siguiendo el orden indicado en el diagrama.

El objetivo de este proceso, es formar los 3 valores iniciales que se han mencionado; dentro de la clase '*systemOfEquations'*. Esto se debe a que dentro de ella es donde se resuelve el análisis, por lo tanto, al definir las variables de esta manera, se evita la necesidad de recurrir a otros objetos para solicitar dicha información.

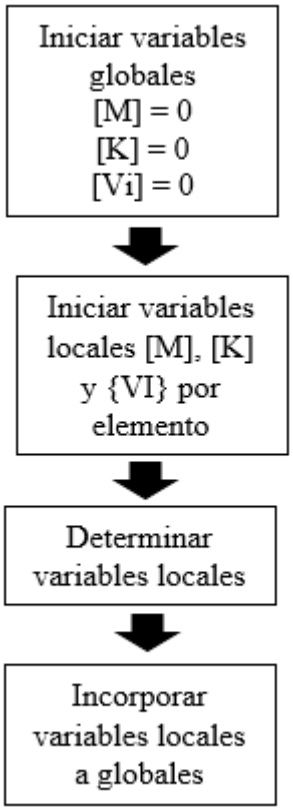

Diagrama 5, diagrama de flujo 'determinar valores iniciales'.

Antes de comenzar a definir las propiedades, se deben obtener los '*nodeHandler'* y '*elementHandler'*. Estos, como se ha mencionado anteriormente, son quienes manejan la información proveniente de los nodos y elementos dentro del análisis. Estos objetos son almacenados en '*analysisModel'*, luego las funciones '*analysisModel.elementHandlerStorage.getComponents'* y

‘*analysisModel.nodeHandlerStorage.getComponents’*, entregan arreglos de tipo ‘*cell’* que contienen los ‘*nodehandler’* y ‘*elementhandler’*, respectivamente.

Una vez determinados, se procede a formar la matriz de rigidez del sistema. Para esto se comienza iniciando de la variable dentro del objeto ‘*systemOfEquation’*, utilizando la función ‘*systemOfEq.zeroK’*. Esta, determina el número de ecuaciones que posee el sistema, es decir, los ‘n’ grados de libertad independientes con los que se cuenta. Luego se guarda, en la propiedad ‘K’, una matriz de ceros de dimensión ‘n x n’. Esto se hace ya que, si no se inicia la variable, cada vez que se agregan términos a esta, se elimina la anterior y se guarda una nueva matriz. Por lo tanto, esto se usa para optimizar el tiempo de trabajo del análisis.

A continuación, se procede con una iteración sobre los ‘*elementHandler’* para determinar su rigidez local e incorporarla a la matriz global. Para esto, se comienza iniciando la rigidez local en cada elemento utilizando la función ‘*elementHandler.zeroTangent’*. Esta función crea una matriz cuadrada de ceros de dimensión igual al número de grados de libertad asociados al elemento.

Luego, se llama la función ‘*elementHandler.addKtToTangent’*. Este método, perteneciente a la clase ‘*elementHandler’*, es el encargado de determinar la matriz de rigidez local de cada elemento y guardarla dentro de las propiedades del objeto. Para esto, debe llamar al elemento propiamente tal, el cual; cuando se creó el objeto ‘*handler’*, fue guardado dentro de sus propiedades.

Cada clase de elemento cuenta, dentro de sus funciones, con un método para determinar su rigidez local, ‘*getTangetStiffness’*. En estas, se define las dimensiones y las propiedades que cumplen la matriz, dependiendo del tipo de elemento que se trate. En las Ecuación 26 y Ecuación 27, se muestra ejemplos de algunos de estos.

$$K_{uniaxial\ Spring} = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix}$$

Ecuación 26, matriz de rigidez elemento ‘uniaxial spring’.

$$K_{beamColumn2D} = \begin{bmatrix} \frac{AE}{L} & 0 & 0 \\ 0 & 4\frac{EI}{L} & 2\frac{EI}{L} \\ 0 & 2\frac{EI}{L} & 4\frac{EI}{L} \end{bmatrix}$$

Ecuación 27, matriz de rigidez elemento 'elastic beam column 2D'.

Finalmente, se ejecuta la función '*systemOfEquation.addTangent'* para agregar la matriz local a la global. A esta, se le entregan, como parámetros de entrada, la matriz local del elemento y el '*ID*' de los grados de libertad asociados a este. Así, la función '*addTangent'*, suma cada componente en la ubicación que le corresponde en la matriz global.

Una vez finalizado el proceso para formar la matriz de rigidez del sistema, se inicia la formación de la matriz de masa. Este, es muy similar al anterior solo que incluye la posibilidad de que existan masas puntuales en los grados de libertad.

El primer paso, es iniciar la variable en la clase '*systemOfEquation'*, con la función '*systemOfEq.zeroM'*. Al igual que en el proceso anterior, se crea una matriz de ceros de dimensión igual al número de grados de libertad independientes en el sistema. Luego, se itera sobre los '*elementHandler'*, en dónde; se inician las variables locales, se determinan las matrices locales y se agrega las componentes a la matriz global de masa de la misma manera que con la de rigidez.

Para formar las matrices locales de masa, se utiliza la función '*elementHanlder.addMToMass'*, la cual trabaja de la misma manera que '*addKtToMass'*; de la misma clase. Esta, obtiene la matriz a partir del elemento con el que se esté trabajando, con la función '*getMass'*. Por lo general, el elemento tributa su masa entre los nodos que lo contienen y determina la inercia que dicha masa genera.

Una vez integradas las matrices locales correspondientes a los elementos, se itera sobre los '*nodeHandler'*. Existe la opción, al momento de definir el sistema, en que se incorporan masas puntuales en los nodos de la estructura. Para esto se debe ingresar, directamente; matriz local del elemento en el orden que se encuentren los grados de libertad, como se muestra en la Ecuación 28. De esta manera, esta se puede agregar a la matriz global siguiendo el orden del '*ID'* del '*nodeHandler'*.

$$M_{masa\ puntual} = \begin{bmatrix} M_x & 0 & 0 & 0 & 0 & 0 \\ 0 & M_y & 0 & 0 & 0 & 0 \\ 0 & 0 & M_z & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x & 0 & 0 \\ 0 & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z \end{bmatrix}$$

Ecuación 28, matriz local de masa para masa puntual.

Por último, se debe definir el vector de influencia. Este indica de qué manera afecta el espectro definido a cada grado de libertad. Como se indicó en el capítulo 3.3.1. Iniciar el análisis, uno de los parámetros de entrada es la dirección en que se desea considerar el espectro. Esta información debe ser consistente con lo que ha establecido el usuario. Por ejemplo, en la Figura 7, se muestran dos típicos sistemas de coordenadas. El de la izquierda, para estructuras en 2D; en donde se considera el desplazamiento en dos dimensiones y el giro en un tercer plano. A la derecha, se muestra un sistema que considera desplazamiento y giro en los tres planos. Luego, la enumeración que se puede ver en las figuras, la define por completo el usuario. Por lo tanto, al momento de definir la dirección de análisis, se debe indicar en dicho orden.

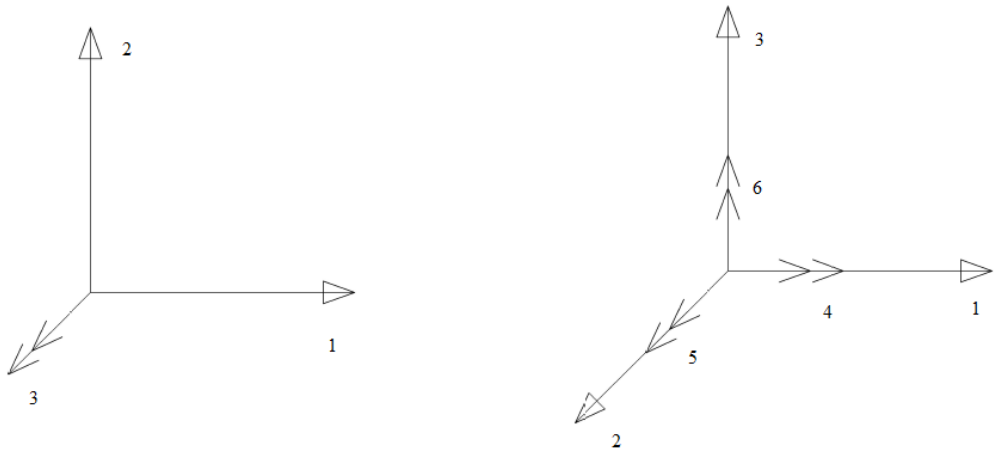

Figura 7, sistemas de coordenadas 2D y 3D.

Para determinar el vector de influencia, se comienza iniciando la variable con la función '*systemOfEquations.zeroVecI'*. Esta crea un vector de ceros, de largo igual al número de grados de libertad independientes del sistema. Luego, se itera sobre los '*nodeHandler'* de la estructura y

ejecutar la función '*systemOfEquations.addInfluence'*. Esta, tiene como parámetros de entrada un 'handler' y la dirección de análisis.

La función '*addInfluence'* tiene dos procedimientos distintitos, dependiendo si alguno de los grados de libertad tiene un '*MP_Constraint'*. En caso de no tener, la función determina el '*ID*' de cada nodo, revisa que el grado de libertad donde se considera el espectro exista y no se encuentre restringido y luego, asigna un '1' en la posición correspondiente del vector '*VecI'*.

En caso de existir un '*MP_Constraint'*, la función reacomoda la variable '*dir'* de tal manera que calce con el nuevo 'ID' del nodo. Más adelante, explicaremos cómo funcionan estas restricciones, pero cabe señalar que se modifica el 'ID' de tal manera que los grados de libertad asociados a la restricción quedan al final. Por lo tanto, al acomodar la dirección, se busca respetar el orden inicial de los grados de libertad para incorporar de manera correcta la influencia de estos al vector '*VecI'*.

De esta manera, se definen los valores iniciales del sistema necesarios para en análisis modal espectral. A continuación, se continúa la explicación de las siguientes etapas del método.

3.3.3. Problema de valores y vectores propios

Como se mostró en los sub capítulos anteriores, ya se ha revisado el modelo y se han generado las variables iniciales del sistema, por lo tanto, se puede continuar con el análisis propiamente tal. Para esto, la primera etapa es resolver el problema de valores y vectores propios. En el Diagrama , se muestra el flujo que sigue este proceso.

Dentro de '*systemOfEquations'*, existe una función encargada de resolver este problema y ajustar las formas modales y las frecuencias del sistema. Esta es la función '*systemOfEquations.solveEigenProblem'* y en este sub capítulo se explica cómo funciona.

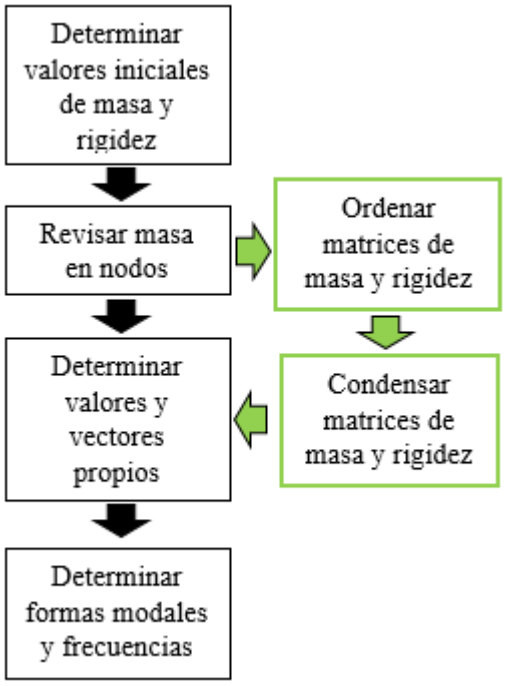

Diagrama 5, diagrama de flujo 'problema de valores y vectores propios'.

En la sección anterior, se explicó por qué las variables de masa, rigidez y vector de influencia, se guardan dentro de '*systemOfEquations'*. El primer paso para resolver el problema de valores y vectores propios es determinar las variables iniciales. Para esto utiliza las funciones '*getK'* y '*getM'* que entregan las matrices de rigidez y masa respectivamente. Si alguna de estas dos es igual a cero, el análisis arroja un mensaje de '*warning'*.

Luego, es necesario revisar si existe masa en todos los grados de libertad y definir los grados activos y pasivos. Esto es muy importante, ya que, si se trabaja con los grados pasivos se indefine el sistema; ya que la matriz de masa contiene ceros en su diagonal. Para solucionar esto, se utiliza la función '*systemOfEquation.hasMass'*, la que revisa si existe masa en todos los grados de libertad. En caso de ser falso, se inicia el proceso para reordenar las matrices.

La función '*SystemOfEquations.order'*, se encarga de ordenar la matriz de masa de tal manera, que los grados de libertad activos queden en la parte superior de la matriz. Para esto, se utiliza el método de pre y post multiplicar por una matriz de identidad, donde se encuentran invertidas las filas y columnas que se desean cambiar.

Además, es posible intercambiar más de una componente haciendo el mismo proceso. A partir de esto, aparecen las variables 'C1' y 'C2', las que guardan la multiplicación de todas las matrices que pre y post multiplican la matriz original respectivamente, cabe mencionar que una es la transposición de la otra. Es necesario guardar estos valores ya que, durante el resto del análisis; los grados de libertad se trabajarán según este nuevo orden, por lo tanto, se deben modificar tanto la matriz de rigidez como el vector de influencia de la misma manera. También, para volver a la distribución original al final del análisis, será necesario invertir el proceso.

Una vez generados los cambios de orden sobre la matriz de rigidez y el vector de influencia. Se obtiene una división entre los grados de libertad activos y pasivos; como se muestra en la Ecuación 29. Estos valores finales, son guardados dentro de las propiedades '*M'*, '*K'* y '*VecI'* de la clase.

$$M = \begin{bmatrix} M_{aa} & M_{ap} \\ M_{pa} & M_{pp} \end{bmatrix}, \qquad K = \begin{bmatrix} K_{aa} & K_{ap} \\ K_{pa} & K_{pp} \end{bmatrix}, \qquad VecI = \begin{bmatrix} VecI_a \\ VecI_p \end{bmatrix}$$

Donde los sub índices representan:

X_{aa}: matriz 'activa activa', representa la influencia de los grados de libertad activos sobre estos mismos.

X_{ap}: matriz 'activa pasiva', representa la influencia de los grados de libertad activos sobre los pasivos.

X_{pa}: matriz 'pasiva activa', representa la influencia de los grados de libertad pasivos sobre los activos.

X_{pp}: matriz 'pasiva pasiva, representa la influencia de los grados de libertad pasivos sobre estos mismos

Ecuación 29, matrices de masa y rigidez y vector de influencia ordenados.

El siguiente paso en el análisis, es condensar las matrices para poder trabajar el problema de valores y vectores propios, para esto se utiliza la función '*systemOfEquations.condesate'*. Lo primero que hace esta función, es determinar el número de grados de libertad activos y extraer la sub matriz de

masa y vector de influencia para los grados de libertad activos. Éstos corresponden a aquellos grados de libertad que mueven alguna masa dentro del sistema.

Por otro lado, se realiza la condensación estática o de Guyan sobre la matriz de rigidez. De esta manera, se obtiene la rigidez de los grados de libertad activos, pero se incluye el aporte de los grados pasivos. Como se indicó luego de explicar la función '*order'*, la matriz de rigidez se encuentra ordenada como se muestra en la Ecuación 29. Bajo esta misma notación, la ecuación de Guyan se define en la Ecuación 30.

$$K = K_{aa} - K_{ap} * {K_{pp}}^{-1} * K_{pa}$$

Ecuación 30, ecuación de Guyan.

Una vez determinados los tres valores condensados, se guardan dentro de las propiedades de la clase como '*Mc'*, '*Kc'* y '*VecIc'*.

De esta manera, se encuentran definidos los valores iniciales para el problema de valores y vectores propios. A continuación, se utiliza la función de MATLAB '*eig'*, como se muestra en la Ecuación 31, esta tiene como parámetros de entrada '$M^{-1} * K$', que corresponden a las matrices condensadas que fueron determinados previamente.

$$[V, D] = eig(M^{-1} * K)$$

Ecuación 31, función 'eig' MATLAB.

Cuando se utiliza la función 'eig' de esta manera, es decir; igualada a '$[V, D]$', esta entrega como resultados una matriz diagonal 'D' de valores propios y una matriz 'V' que corresponde a los vectores propios del sistema. Luego, se determina la parte real de ambas componentes y se obtiene la raíz cuadrada de la diagonal de 'D'. Estos valores finales, corresponden a la matriz de forma modal 'φ' y las frecuencias naturales del sistema 'ω'. Las cuales son guardadas dentro de las propiedades del objeto como '*eigenVector'* y '*eigenValue'* respectivamente.

3.3.4. Análisis modal espectral

Esta, es la etapa principal del análisis modal espectral, ya que aquí; se efectúa el análisis en cuestión. En el Diagrama 6, se muestran los principales pasos que se explican en esta sección. Todo

este proceso se realiza en la función '*systemOfEquations.solve'*, que al igual que la función descrita en el paso anterior; se efectúa dentro de la función 'analyze' de '*eigenAnalysis'*. Esta, solo tiene como parámetro de entrada el espectro que fue definido con el objeto '*eigenAnalysis'*.

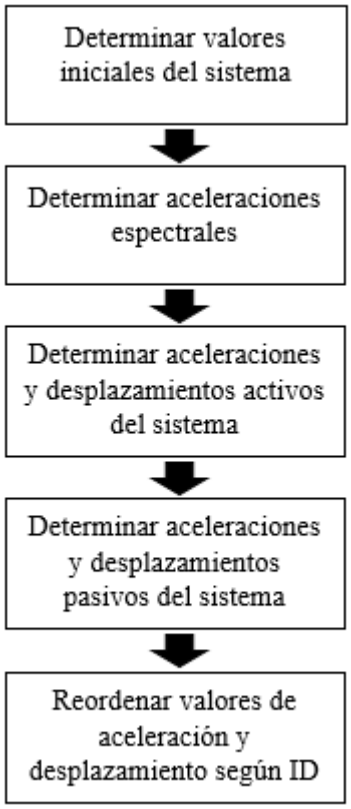

Diagrama 6, diagrama de flujo 'análisis modal espectral'.

Antes de definir los valores iniciales, es necesario saber si existen grados de libertad pasivos en el sistema original. Y al igual que en el paso anterior, se utiliza la función '*systemOfEquations.hasMass'*. En caso de entregar 'true', se utiliza la matriz de masa y el vector de influencia condensados, sino; los valores regulares. De cualquier manera, estos son determinados a partir de las propiedades de la clase. También se obtienen de estas propiedades; los valores de la matriz modal y las frecuencias naturales del sistema.

Luego, se determina el factor de participación modal que representa la amplitud que tendrá la oscilación del sistema. En este tipo de análisis, el factor se determina como se muestra en la

Ecuación 32, de la que se obtiene un vector de longitud igual al número de grados de libertad activos.

$$FPM = \frac{\{\varphi\}^T{}_i * [M] * [VI]}{\{\varphi\}^T{}_i * [M] * \{\varphi\}_i}$$

Ecuación 32, factor de participación modal.

Junto con esto, se inicia la variable 'Ra', en la que luego serán definidas las aceleraciones espectrales. Para determinar dichas aceleraciones se utiliza la función '*systemOfEquation.spectralAccel'*. A esta, se le entregan de a uno los periodos de la estructura y el espectro de diseño, luego determina la aceleración correspondiente a dicho periodo interpolando entre los valores del espectro. Junto con esto, se determinan los desplazamientos modales '*Rd'*, los cuales se obtienen al dividir las aceleraciones por las frecuencias naturales al cuadrado. Este concepto surge a partir del pseudo espectro de desplazamiento que se ha explicado en el capítulo 2.

A continuación, se calculan los desplazamientos y aceleraciones para cada grado de libertad y para cada modo. Para esto se utilizan las fórmulas de la Ecuación 33, de donde se obtiene una matriz como la de la Ecuación 34, separando el resultado por modo y por grado de libertad.

$$Y_{aceleracion} = \{\varphi\}^T{}_i * FPM_i * Ra_i$$

$$Y_{desplazamiento} = \{\varphi\}^T{}_i * FPM_i * Rd_i$$

Ecuación 33, desplazamientos y aceleraciones modales del sistema.

$$Y_{aceleración} = \underbrace{\begin{bmatrix} Ya_{11} & \dots & Ya_{n3} \\ \vdots & \ddots & \vdots \\ Ya_{1n} & \dots & Ya_{nn} \end{bmatrix}}_{modo\ 1 \quad \dots \quad modo\ n} \begin{matrix} gdl\ 1 \\ \vdots \\ gdl\ n \end{matrix}$$

Ecuación 34, matriz de aceleración modal.

Como anteriormente se condensaron las matrices de masa y rigidez del sistema, los valores que se obtienen con la Ecuación 33, corresponden exclusivamente a los desplazamientos y aceleraciones de los grados de libertad activos. En efecto, es necesario determinar la solución para los grados pasivos en caso que estos existan.

Para esto, se comienza iniciando las variables '*dispByMode'* y '*accelByMode'*, como matrices de ceros de dimensión igual al número de grados de libertad independientes por el número de modos de la estructura. La cantidad de grados independientes, se determina a partir de la propiedad '*numberOfEquetions'* y los modos a partir de la cantidad de frecuencias naturales.

A continuación, se itera modo a modo, de tal manera que se define la aceleración y desplazamiento activo a partir de cada columna de la matriz '*Ya'* y '*Yd'*. Luego, utilizando las fórmulas que se muestran en la Ecuación 35, se determinan los valores para las aceleraciones y desplazamientos pasivos.

$$aceleración_{pasiva} = -K_{pp}^{-1} * K_{ap} * aceleración_{activa}$$

$$desplazamiento_{pasiva} = -K_{pp}^{-1} * K_{ap} * desplazamiento_{activa}$$

Ecuación 35, aceleración y desplazamiento pasivos.

Finalmente, ambos resultados se concatenan y se pre multiplicados por las matrices de orden que fueron definidas al resolver el problema de valores y vectores propios. De esta manera, se recupera el orden inicial de los grados de libertad del sistema que se indica en el '*ID*' de cada nodo.

Estas matrices finales, son guardadas en las propiedades '*dispByMode'* y '*accelByMode'* de '*systemOfEquations'*. Es importante conservar estos valores; sin aplicar la combinación, ya que en el caso de la '*CQC'*, se trata de un método no lineal. Luego, si se necesitan estos valores para determinar otros resultados, se debe aplicar el procedimiento modo a modo y luego hacer la combinación.

Una vez resuelto el análisis modal espectral, es necesario incorporar los valores de la matriz de formas modales a cada nodo. Esto se utiliza luego, para determinar los esfuerzos en los elementos y para conocer las deformaciones modales de la estructura

Como cada modo de vibrar puede estar normalizado por una constante, se puede utilizar la matriz de desplazamientos modales, en donde los vectores propios están multiplicados por el factor de participación modal y un desplazamiento espectral.

$$Y_{desp} = [\varphi] * FPM * Ra$$

Ecuación 36, matriz modal amplificada.

Como se vio anteriormente, la matriz 'Y_{desp}'; contiene el desplazamiento todos los modos de vibrar y todos los grados de libertad independientes. Por lo tanto, solo falta asignar nodo a nodo los valores según su '*ID'* y para todos los modos de vibrar.

Para determinarlos, se itera sobre los '*nodeHandler'*. En esta iteración, se determina el '*ID'* del *ntransformationMatrix'odo*, el '*transformationID'* y la ', estos conceptos se explicarán en el siguiente capítulo, IMPLEMENTACIÓN DE LA RESTRICCIÓN DE MÚLTIPLES NODOS TIPO DIAGRAGMA RÍGIDO.

Luego, se obtiene la sub matriz de las formas modales correspondientes según los grados de libertad para cada nodo. Para esto, se extraen las filas correspondientes indicadas en el '*transID'*, como se muestra en la Ecuación 37. Aquí, la matriz de desplazamientos modal posee 'n' modos de vibrar y 'm' grados de libertad independientes. Luego, el nodo 'x', tiene asociados los grados de libertad 'm', 'n' y 'p', por lo que extrae esas filas.

$$\varphi = \begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1n} \\ \vdots & \ddots & \vdots \\ \varphi_{p1} & \cdots & \varphi_{pn} \\ \vdots & \ddots & \vdots \\ \varphi_{q1} & \cdots & \varphi_{qn} \\ \vdots & \ddots & \vdots \\ \varphi_{r1} & \cdots & \varphi_{rn} \\ \vdots & \ddots & \vdots \\ \varphi_{m1} & \cdots & \varphi_{mn} \end{bmatrix}$$

$$trans\ ID_{nodo\ x} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$$\varphi_{nodo\ x} = \begin{bmatrix} \varphi_{p1} & \cdots & \varphi_{pn} \\ \varphi_{q1} & \cdots & \varphi_{qn} \\ \varphi_{r1} & \cdots & \varphi_{rn} \end{bmatrix}$$

Ecuación 37, determinar sub matriz de desplazamientos modales

Para terminar, se multiplica esta última matriz por la matriz de transformación y se obtienen los desplazamientos modales para los grados de libertad indicados en el 'ID' del nodo. Este valor, se ajusta en el nodo vector a vector, utilizando la función '*node.setEigenVector'*, la que recibe como información de entrada el vector y el número al que este corresponde.

De esta manera, finaliza esta etapa del desarrollo del análisis modal espectral. Las secciones que se presentan a continuación trabajan en base a la información que se ha generado en este proceso y buscan entregar de manera más ordenada y completa la información al usuario.

3.3.5. Determinar fuerzas resistentes

Las fuerzas resistentes corresponden a las fuerzas que se generan en cada grado de libertad producto del equilibrio estático o dinámico. Estos, al estar asociados a los elementos; le transmiten esas fuerzas y se generan los esfuerzos internos.

El proceso para determinar las fuerzas resistentes se realiza dentro de la función '*analyze'*, pero los resultados obtenidos se almacenan dentro del cada elemento. Por lo tanto, si el usuario desea obtener el valor de estas fuerzas, debe recurrir directamente a la propiedad '*modalReisistingForce'* de la clase madre de todos los elementos. Este proceso, se muestra en el Diagrama 7.

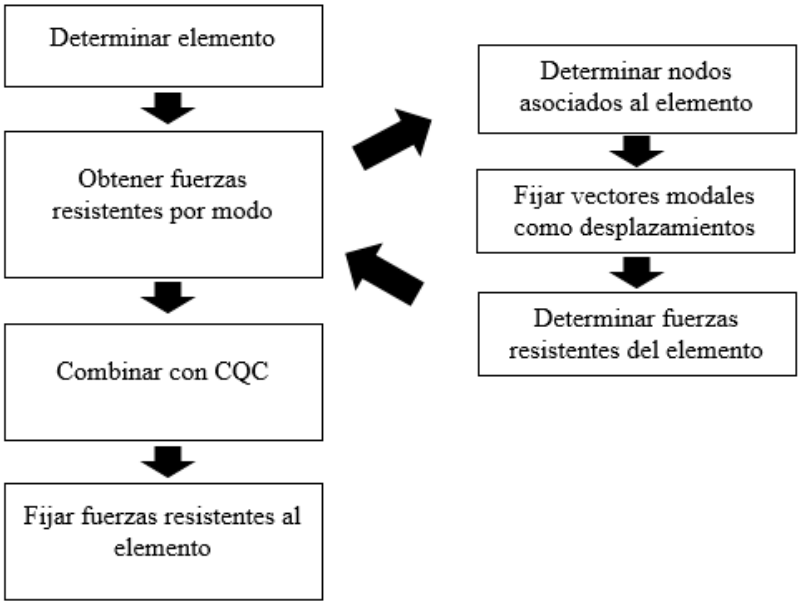

Diagrama 7, diagrama de flujo 'determinar esfuerzos internos'.

Para determinar las fuerzas resistentes, se utiliza en una iteración sobre los '*elementHandlers'*. Primero, se debe obtener el elemento mediante la función '*elementHandler.getElement'*. Esta, entrega el elemento asociado al '*handler'* y permite el acceso a sus propiedades y funciones.

Luego, se ejecuta la función '*element.getModalResistingForce'*. Esta fue agregada como parte de este trabajo de título y solo se utiliza para la obtención de las fuerzas resistentes del análisis modal espectral.

La función no recibe ninguna información de entrada, sino que adquiere información a través de los nodos que se encuentran asociados al elemento. De esta manera, define el número de modos y de nodos, e inicia la variable '*modalResistingForces'* como una matriz de ceros; de dimensión igual al número de grados de libertad asociados al elemento por el número de modos de la estructura.

Luego, itera sobre estos modos, asignando como desplazamientos el vector propio que fue asignado a cada nodo. Finalmente determina los esfuerzos en el elemento mediante la función '*getResistingForces'* y lo guarda en la matriz '*modalResistingForces'* en la columna correspondiente al número del modo de la iteración.

Dentro de la función '*analyze'* y la iteración de los '*elementHandler'*, se procede a combinar con CQC la matriz obtenida con la función anteriormente descrita. Para esto se utiliza la función '*modalSystemOfEquations.CQC'* y se obtienen los esfuerzos combinados del elemento. Estos son guardados en la propiedad '*modalResistingForce'* del elemento con la función '*element.setModalResistingForce'*.

De esta manera, quedan definidos las fuerzas resistentes para cada elemento y guardada dentro de sus propiedades. Por lo tanto, si el usuario requiere esta información la puede obtener con la función '*element.getModalResistingForce'* una vez concluido el análisis.

3.3.6. Determinar desplazamientos

Esta es la etapa final del análisis modal espectral. Aquí se ordenan los valores obtenidos en los pasos anteriores y se ajustan en el modelo, de manera que se permite una sencilla adquisición de resultados para el usuario. En el Diagrama 8, se muestran los pasos que se siguen para el desarrollo de esta etapa del análisis.

Se puede ver que comienza con dos etapas en paralelo. Estas, corresponde a iniciar las variables transformadas y las regulares. El concepto de variables transformadas, se explica en detalle en el siguiente capítulo de IMPLEMENTACIÓN DE LA RESTRICCIÓN DE MÚLTIPLES NODOS TIPO DIAGRAGMA RÍGIDO, mientras que las variables regulares corresponden a los grados de libertad definidos por el usuario al determinar el sistema. Por ahora nos basta saber que, para obtener los desplazamientos finales, se deben obtener los valores transformados y aplicar la transformación de manera inversa.

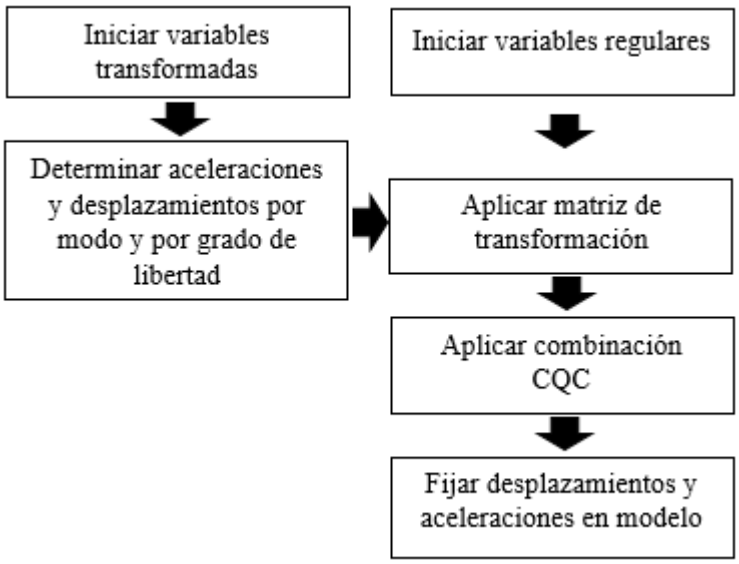

Diagrama 8, diagrama de flujo 'determinar desplazamientos'.

Primero que todo, se debe comenzar una iteración sobre los '*nodehandler'*, de esta manera se puede trabajar con cada uno de ellos de manera independiente. A continuación, se deben adquirir las variables transformadas, que corresponden al '*numberDOFTrans'*; la cual indica el número de grados de libertad asociados al nodo transformado y el '*transID'*; un vector igual que el 'ID' para el nodo transformado. Por otro lado, se deben adquirir las propiedades regulares del sistema. Estas son; el nodo regular y el número de grados de libertad de este.

En seguida, se realiza una iteración entre los modos de la estructura dentro del '*nodeHandler'* en que estemos trabajando. En esta, se comienza completando los desplazamientos y aceleraciones modales que le correspondan según el '*transID'*. Luego se revisa si existe '*MP_Constraint'*, en caso de ser así; se multiplica la matriz de desplazamientos y aceleraciones modales por la matriz de transformación y se obtiene la matriz regular de desplazamientos y aceleraciones modales.

Finalmente, se le aplica la combinación a cada matriz y se obtienen los dos vectores deseados; desplazamiento y aceleración. Estos son incorporados al modelo mediante la función '*node.setTrialDisp'* y '*node.setTrialAccel'*, las que cambian el '*trialState'* de la estructura, para luego; actualizarlo a '*commitState'* con la función '*node.commitState'*. De esta manera el usuario puede obtener el desplazamiento y la aceleración final de cada nodo con la función '*node.getDisp'* y '*node.getAccel*.

CAPÍTULO IV

IMPLEMENTACIÓN DE LA RESTRICCIÓN DE MÚLTIPLES NODOS TIPO DIAGRAGMA RÍGIDO

4.1. Introducción

En este capítulo se presenta la implementación de los diafragmas rígidos en SAFE_TB. Como se explicó anteriormente, los diafragmas rígidos, son un tipo de restricción que genera una relación entre los grados de libertad de dos nodos. Para ser más específico, crea una relación entre los desplazamientos en un plano y el giro ortogonal a este; del nodo maestro y nodo esclavo.

Como en el capítulo de REVISIÓN BIBLIOGRÁFICA, ya se explicó el concepto teórico detrás de este tipo de restricción, este capítulo contiene el desarrollo de la clase '*rigidDiaphragmConstraint'*. Se explica la estructura que esta posee, las funciones principales, la relación que tiene con el resto del programa y como se implementan estas restricciones en el modelo.

4.2. Estructura de la clase '*ridigDiaphragmConstraint'*

Para tener una idea más clara y poder entender mejor cómo funciona la clase que genera diafragmas rígidos, se comenzará explicando la estructura que esta tiene. En esta parte del capítulo se muestra como se generan objetos de esta clase, cómo funciona el constructor y cuáles son sus funciones principales.

Para comenzar, es necesario comprender que la clase '*rigidDiaphramgConstraint'* es sub clase de la clase '*MP_Constraint'*, por lo tanto, existen propiedades y métodos que ambas comparten producto de la herencia, mientras que otras corresponden solo a la sub clase.

Al momento de definir un objeto '*MP_Constraint'*, se le debe asignar una etiqueta, un nodo maestro y uno esclavo, los grados de libertad que se encuentran relacionados y una matriz de restricción. Los grados que se encuentran relacionados; se dividen en dos grupos, aquellos que le pertenecen al nodo maestro, que llamaremos grados maestros y los que pertenecen al nodo esclavo o nodos esclavos.

Por otro lado, en un diafragma rígido solo se indica su etiqueta y los nodos maestro y esclavo. Esto se debe, a que la relación entre los grados de libertad maestros y esclavos; esta predeterminada.

De esta manera, en el constructor de la clase hija se define un objeto de la clase base con elementos de entrada determinados. A este se le entregan la misma etiqueta y nodos con que fue definido el diafragma, pero los grados de libertad asociados a la restricción son siempre los mismos. Estos, se muestran en verde en la Ecuación 38 y corresponden a los grados que fueron indicados en el capítulo de revisión bibliográfica. También se le asigna una matriz de transformación vacía, ya que esta es formada más adelante.

$$y_{nodo\ maestro} = \begin{Bmatrix} r_x \\ r_y \\ r_z \\ \vartheta_x \\ \vartheta_y \\ \vartheta_z \end{Bmatrix}, y_{nodo\ esclavo} = \begin{Bmatrix} r_x \\ r_y \\ r_z \\ \vartheta_x \\ \vartheta_y \\ \vartheta_z \end{Bmatrix}$$

Ecuación 38, grados de libertad sometidos a la restricción de diafragma rígido.

Una vez que se encuentra creado el objeto de la restricción, se procede a agregarlo al modelo mediante la función '*model.addMP_Constraint'*. En esta se ejecuta la función '*rigidDiaphragmConstraint.setModel'*. Ésta crea la matriz de transformación, la cual fue explicada en el capítulo REVISIÓN BIBLIOGRÁFICA y representa la relación entre los grados de libertad de los nodos.

Otras de las funciones que posee esta clase son; '*delete'*, *disp'* y funciones '*get'* y '*set'*, que se trabajan de la misma manera que otras que ya se han explicado anteriormente. Cabe mencionar que la función '*disp'* entrega información de; los nodos esclavo y maestro y los grados de libertad restringidos en cada uno.

4.3. Funcionamiento de la clase '*ridigDiaphragmConstraint*'

A continuación, se explica cómo, las partes mencionadas en la sección anterior, se relacionan con el resto del programa. Con esto se busca tener una comprensión global de cómo trabaja esta clase y como asigna restricción a los nodos asociados.

La clase de diafragmas rígidos, tiene influencia sobre el '*ID*' y la matriz de transformación del '*nodeHandler'* asociado al nodo esclavo de la restricción. De esta manera, influye sobre la formación de las matrices de masa y rigidez y sobre el vector de influencia y sobre la solución del sistema. Su acción sobre el 'ID', se genera mediante las funciones de '*constraintHandler.handle'*, '*DOFNumberer.numberDOF'* y '*constraintHandler.doneNumberingDOF'*, las que se ejecutan dentro de la función '*eigenAnalysis.modelChanged'*.

La función '*constraintHandler.handle'*, es la encargada de generar los '*nodeHandler'* y '*elementHandler'*. Este proceso, comienza con una iteración sobre los nodos almacenados en el modelo. Para cada uno de ellos, se define un '*nodeHandler'*, el cual es definido como '*nodeTransformationHandler'* en caso que el nodo este asociado a cualquier tipo de restricción.

Luego, para cada uno de estos, se define la variable 'ID' como un vector de ceros de dimensión igual al número de grados de libertad que posea el nodo. Este vector, es completado con un '-1', en la posición correspondiente para cada grado de libertad que se encuentre sometido a una restricción simple. En el caso de las restricciones múltiples, se incorpora un '-2' en los grados de libertad esclavos, para luego ser reordenada, de tal manera; que los grados restringidos queden al final del vector.

Dentro de esta iteración, se determina el número de ecuaciones del sistema. Este valor corresponde el número total de grados de libertad independientes y se utiliza en todos los tipos de análisis. Para determinarlo, se suman los grados de libertad de cada nodo y se le resta el número de grados que se encuentren asociados a alguna restricción.

Después de este proceso, el 'ID' queda compuesto por ceros y los valores que indican una restricción. En el caso de los diafragmas rígidos, los grados de libertad maestros y esclavos son 3, al igual que los grados independientes. Por lo tanto, el 'ID' resultante queda como se muestra en la Ecuación 39.

$$ID_{nodo\ esclavo} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -2 \\ -2 \\ -2 \end{bmatrix}$$

Ecuación 39, 'ID' en diafragmas rígidos luego de 'constraintHandler.handle'.

Por otro lado, se itera sobre todos los elementos del modelo y al igual que en la iteración anterior, se genera un '*elementHandler'* para cada uno. Si alguno de los grados de libertad del elemento cuenta con alguna restricción, el '*handler'* se generan mediante la clase '*elementTransformationHandler'*.

A continuación, se utiliza la función '*DOFNumberer.numberDOF'*, que es la encargada de asignar el etiquetado único en el ID. Como se ha mencionado anteriormente, el '*ID'* es la propiedad que le permite al programa localizar los grados de libertad durante el análisis. De esta manera, se logra trabajar integrando las matrices locales a la global y determinar cuáles de los resultados finales corresponden a que nodo y a que elemento.

El modo de operar de esta función es mediante un contador, el cual comienza en 1. Luego, cada vez que asigna el número del contados a un grado de libertad, le suma uno. Cabe señalar, que los espacios que se encuentran ocupados con un '-1' o un '-2' no son llenados, ya que estos no corresponden a un grado de libertad del sistema.

$$ID_{nodo\ esclavo} = \begin{bmatrix} ID_{esclavo\ 1} \\ ID_{esclavo\ 2} \\ ID_{esclavo\ 3} \\ -2 \\ -2 \\ -2 \end{bmatrix}$$

Ecuación 40, 'ID' en diafragmas rígidos luego de 'DOFNumberer.numberDOF'.

Finalmente, la función '*constraintHandler.doneNumberingDOF'*, itera sobre los '*nodeHandler'* llamando a la función '*nodeHandler.doneID'*, y sobre los '*elementHandler'* llamando a '*elementHandler.setID'*.

La primera, es la encargada de enumerar los espacios del 'ID' que fueron llenado con '-2', es decir, aquellos que pertenecen a algún '*MP_Constraint'*. Esta, solo opera si existe un '*MP_Constraint*, por lo tanto, lo primero que hace es revisarlo. Luego, asigna el 'ID' de los grados de liberad maestros, en los espacios correspondientes en el nodo esclavo. El orden en que estos son incorporados es muy importante, ya que debe ser igual al orden impuesto cuando se formó la matriz

de transformación. De esta manera, se obtiene la relación deseada entre los grados de libertad esclavos y maestros.

$$ID_{nodo\ esclavo} = \begin{bmatrix} ID_{esclavo\ 1} \\ ID_{esclavo\ 2} \\ ID_{esclavo\ 3} \\ ID_{maestro\ 1} \\ ID_{maestro\ 2} \\ ID_{maestro\ 3} \end{bmatrix}$$

Ecuación 41, 'ID' en diafragmas rígidos luego de 'constraintHandler.doneNumberingDOF'.

Por otro lado, la función '*elementHandler.setID'*; incorpora los valores del 'ID', de todos los nodos asociados al elemento, en el '*elementHandler'*. De esta manera, se forma el 'ID' del elemento, que trabaja de la misma manera que el de los nodos. También se forma la matriz de transformación del elemento, la cual es una unión de las matrices de cada nodo en el mismo orden en que se concatenó el 'ID'.

Una vez determinado el 'ID' para todos los nodos y elementos, el análisis continúa de forma normal. Pero cuando el sistema forme las matrices de masa y rigidez y el vector de influencia, como se explicó en el capítulo anterior, incluirá el efecto de los elementos con diafragma rígido de otra manera.

También al determinar la solución, el análisis solo determina la respuesta de los grados de libertad maestros asociados al diafragma. La respuesta de los grados de libertad esclavos, se determinan a partir de la relación que existe entre ambos mediante la matriz de transformación, como se muestra en la Ecuación 42.

$$ID_{nodo\ escl}\begin{bmatrix} ID_{maestro\ 1} - ID_{maestro\ 3} * d_y \\ ID_{maestro\ 2} + ID_{maestro\ 3} * d_x \\ ID_{esclavo\ 1} \\ ID_{esclavo\ 2} \\ ID_{esclavo\ 3} \\ ID_{maestro\ 3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & -d_y \\ 0 & 0 & 0 & 0 & 1 & d_x \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} ID_{esclavo\ 1} \\ ID_{esclavo\ 2} \\ ID_{esclavo\ 3} \\ ID_{maestro\ 1} \\ ID_{maestro\ 2} \\ ID_{maestro\ 3} \end{bmatrix}$$

Ecuación 42, efecto de la matriz de transformación.

CAPÍTULO V

PRESENTACIÓN Y EVALUACIÓN DE RESULTADOS

5.1. Introducción

En este capítulo, se muestran los resultados obtenidos en una serie de modelo con distintas propiedades. Estos, fueron analizados mediante el método modal espectral en las plataformas SAFE_TB y SAP2000. De esta manera se busca generar un punto de comparación entre el análisis desarrollado en este trabajo de título y otro con licencia comercial.

Se determinó realizar la comparación utilizando el programa SAP2000 por varios motivos. Primero que todo, la Universidad de Chile cuenta con licencias de este programa en el departamento de ingeniería civil, de esta manera se puede usar sin problemas. También, existe una similitud entre ambas plataformas. Las dos trabajan en base a elementos finitos, cuentan con una interfaz gráfica y su estructura está orientadas a objetos. Finalmente, SAP2000, junto a los demás programas de CSI, son de los programas más utilizados en Chile para el diseño estructural. De esta manera, se facilita el proceso de aprendizaje; ya que existe una gran cantidad de información al respecto.

Las estructuras que se desarrollan en este capítulo están divididas en dos grupos; estructuras en 2D y en 3D. Durante la implementación del análisis modal espectral y la restricción de diafragmas rígidos, se comenzó trabajando con modelos simples, para luego; ir aumentando el nivel de complejidad de la estructura a analizar.

A continuación, se presentan los modelos para la comparación en este mismo orden. En este capítulo, se explica cómo está definido el modelo y las consideraciones que se tiene en cada uno. También, como están asignados los conceptos que se han revisado en los capítulos anteriores y por supuesto; los resultados obtenidos en ambas plataformas.

5.2. Estructuras en 2D

5.2.1. Marco 2D de un piso

Número de grados de libertad: 6
Número de elementos: 3

Tipo de elementos: ElasticBeamColumn2D

Apoyos: 2 (empotrados)

Diafragmas rígidos: No

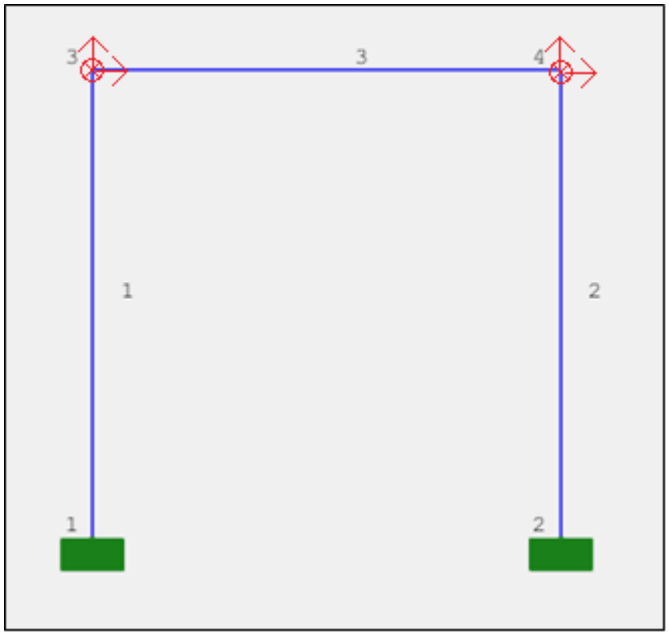

Figura 8, Modelo 3 marco 2D con masa puntual.

Este modelo, corresponde a un marco típico, en donde sus nodos poseen 3 grados de libertad. Los inferiores se encuentran empotrados y los superiores poseen una masa puntual de 0,1 [ton]. Sus dimensiones son; 3 [m] de altura y 3 [m] de ancho. Los elementos 'elasticBeamColumn2D' consideran un perfil 'rectangularProfile' de 0,1 x 0,1 [m^2] y un material 'elasticUniaxilMaterial' con un módulo de elasticidad igual a 20.000.000 [tonf/m^2]. El sismo se considera en la dirección horizontal y el espectro con el que se evaluará el análisis modal espectral; corresponde a la zona 1 y suelo A, según la norma NCh 433.

Los grados de libertad del sistema corresponden al desplazamiento vertical, desplazamiento horizontal y el giro fuera del plano en los nodos 3 y 4.

Los códigos de la implementación de este modelo para SAFE_TB, se encuentran en el Anexo 1.

Resultados:

A continuación, se muestran los resultados obtenidos a partir del modelo que se muestra en la Figura 8. En cada tabla, se presentan los valores obtenidos en ambas plataformas y el porcentaje de error que existe entre estos.

	Periodo [seg]		
	SAFE_TB	SAP2000	ξ
Modo 1	0,2760	0,2765	-0,17%
Modo 2	0,0077	0,0077	0,06%
Modo3	0,0077	0,0077	0,10%
Modo 4	0,0054	0,0054	-0,75%

Tabla 1, Periodos modelo 1.

	Formar modales normalizadas			
	SAFE_TB			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	1	0	0	-1
GDL 2	0	1	-1	0
GDL 3	1	0	0	1
GDL 4	0	1	1	0

	SAP2000			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	1	0	0	-1
GDL 2	0	1	-1	0
GDL 3	1	0	0	1
GDL 4	0	1	1	0

	ξ			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	0,00%	0,00%	0,00%	0,00%
GDL 2	0,00%	0,00%	0,00%	0,00%
GDL 3	0,00%	0,00%	0,00%	0,00%
GDL 4	0,00%	0,00%	0,00%	0,00%

Tabla 2, Modos de vibrar normalizadas modelo 1.

	Desplazamientos por nodo Análisis Modal Espectral		
	SAFE_TB		
	Ux	Uy	Θy
Nodo 1	0	0	0
Nodo 2	0	0	0
Nodo 3	0,0077	0	0,0015
Nodo 4	0,0077	0	0,0015

	SAP2000		
	Ux	Uy	Θy
Nodo 1	0	0	0
Nodo 2	0	0	0
Nodo 3	0,0077	0	0,0015
Nodo 4	0,0077	0	0,0015

	ξ		
	Ux	Uy	Θy
Nodo 1	0,00%	0,00%	0,00%
Nodo 2	0,00%	0,00%	0,00%
Nodo 3	-0,22%	0,00%	0,00%
Nodo 4	-0,22%	0,00%	0,00%

Tabla 3, desplazamientos análisis modal espectral modelo 1.

		Fuerzas resistentes Análisis Modal Espectral		
		SAFE_TB		
Elemento		Fx	Fz	My
1	Start	0,3965	0,3400	0,6798
1	End	0,3965	0,3400	0,5096
2	Start	0,3965	0,3400	0,6798
2	End	0,3965	0,3400	0,5096
3	Start	0	0,3397	0,5096
3	End	0	0,3397	0,5096

		SAP2000		
Elemento		Fx	Fz	My
1	Start	0,3960	0,3394	0,6793
1	End	0,3960	0,3394	0,5087
2	Start	0,3960	0,3394	0,6793
2	End	0,3960	0,3394	0,5087
3	Start	0	0,3392	0,5087
3	End	0	0,3392	0,5087

		ξ		
Elemento		Fx	Fz	My
1	Start	0,12%	0,17%	0,08%
1	End	0,12%	0,17%	0,17%
2	Start	0,13%	0,18%	0,08%
2	End	0,13%	0,18%	0,17%
3	Start	0,00%	0,15%	0,17%
3	End	0,00%	0,15%	0,17%

Tabla 4, fuerzas resistentes análisis modal espectral modelo 1.

5.2.2. Marco 2D de un piso con masa distribuida

Número de grados de libertad: 6

Número de elementos: 3

Tipo de elementos: BeamColumn2D

Apoyos: 2 (empotrados)

Diafragmas rígidos: No

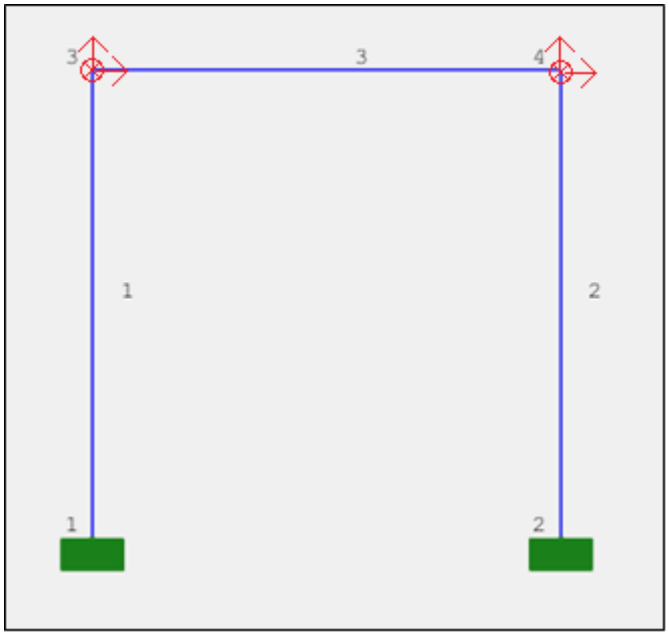

Figura 9, Modelo 2 marco 2D.

En este modelo posee la misma geometría, elementos, módulo de elasticidad y espectro de diseño que el anterior. La diferencia entre ambos, es que éste considera una masa distribuida de 2 [ton/m^3] en vez de las masas puntuales. Esto que genera un cambio en la matriz de masa ya que se deben integrar las matrices locales.

Los grados de libertad del sistema corresponden al desplazamiento vertical, desplazamiento horizontal y el giro fuera del plano en los nodos 3 y 4.

Los códigos de la implementación de este modelo para SAFE_TB, se encuentran en el Anexo 2.

Resultados:

	Periodo [seg]		
	SAFE_TB	SAP2000	ξ
Modo 1	0,4275	0,4277	-0,04%
Modo 2	0,0060	0,0060	0,00%
Modo3	0,0060	0,0060	0,00%
Modo 4	0,0042	0,0042	0,00%

Tabla 5, periodos modelo 2.

	Formar modales normalizadas			
	SAFE_TB			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	1	0	0,0002	1
GDL 2	0,0002	1	-1	0
GDL 3	1	0	0,0002	-1
GDL 4	-0,0002	1	1	0

	SAP2000			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	1	0	0,0002	1
GDL 2	0,0002	1	-1	0
GDL 3	1	0	0,0002	-1
GDL 4	-0,0002	1	1	0

	ξ			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	0,00%	0,00%	0,00%	0,00%
GDL 2	0,00%	0,00%	0,00%	0,00%
GDL 3	0,00%	0,00%	0,00%	0,00%
GDL 4	0,00%	0,00%	0,00%	0,00%

Tabla 6, modos de vibrar normalizados modelo 2.

	Desplazamientos por nodo Análisis Modal Espectral		
	SAFE_TB		
Nodo 1	0	0	0
Nodo 2	0	0	0
Nodo 3	0,0127	0	0,0026
Nodo 4	0,01273	0	0,00255

	SAP2000		
	Ux	Uy	θy
Nodo 1	0	0	0
Nodo 2	0	0	0
Nodo 3	0,0127	0	0,0025
Nodo 4	0,0127	0	0,0025

	ξ		
	Ux	Uy	θy
Nodo 1	0,00%	0,00%	0,00%
Nodo 2	0,00%	0,00%	0,00%
Nodo 3	-0,02%	0,00%	0,12%
Nodo 4	-0,02%	0,00%	0,12%

Tabla 7, desplazamientos análisis modal espectral modelo 2.

		Fuerzas resistentes Análisis Modal Espectral		
		SAFE_TB		
Elemento		Fx	Fz	My
1	Start	0,03535	0,04124	0,07071
1	End	0,03535	0,04124	0,05302
2	Start	0	0,0354	0,05302
2	End	0	0,0354	0,05302
3	Start	0,03535	0,04124	0,05302
3	End	0,03535	0,04124	0,07071

		SAP2000		
Elemento		Fx	Fz	My
1	Start	0,0353	0,0412	0,0707
1	End	0,0353	0,0412	0,0530
2	Start	7,97E-16	0,0353	0,0530
2	End	7,97E-16	0,0353	0,0530
3	Start	0,0353	0,0412	0,0530
3	End	0,0353	0,0412	0,0707

		ξ		
Elemento		Fx	Fz	My
1	Start	0,14%	0,10%	0,03%
1	End	0,14%	0,10%	0,04%
2	Start	0,00%	0,14%	0,04%
2	End	0,00%	0,14%	0,04%
3	Start	0,14%	0,10%	0,04%
3	End	0,14%	0,10%	0,03%

Tabla 8, fuerzas resistentes análisis modal espectral.

5.2.3. Marco 2D de tres pisos con masa distribuida

Número de grados de libertad: 24

Número de elementos: 12

Tipo de elementos: BeamColumn2D

Apoyos: 2 (empotrados)

Diafragmas rígidos: No

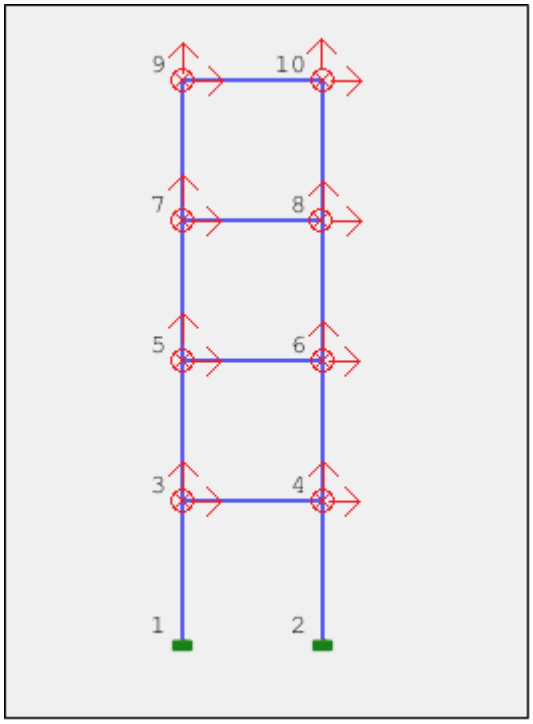

Figura 10, Modelo 4 marco 2D 4 niveles.

Este modelo tiene iguales propiedades al anterior cuenta con 4 niveles. De esta manera se extiende la comparación entre las plataformas al aumentar el número de grados de libertad a comprar.

La estructura cuenta con dos apoyos en los nodos inferiores los cuales son restringidos en sus 3 grados de libertad. En total, la estructura cuenta con 16 grados independientes.

Los grados de libertad del sistema corresponden al desplazamiento vertical, desplazamiento horizontal y el giro fuera del plano en los nodos 3, 4, 5, 6, 7, 8, 9 y 10.

Los códigos de la implementación de este modelo para SAFE_TB, se encuentran en el Anexo 3.

Resultados:

	Periodo [seg]		
	SAFE_TB	SAP2000	ξ
Modo 1	1,2681	1,26857	-0,04%
Modo 2	0,4047	0,404794	-0,04%
Modo3	0,2339	0,233945	-0,04%
Modo 4	0,1717	0,171749	-0,04%
Modo 5	0,0138	0,01378	0,00%
Modo 6	0,0138	0,013762	-0,01%
Modo 7	0,0048	0,004835	0,10%
Modo 8	0,0048	0,004834	-0,08%
Modo 9	0,0037	0,00365	0,00%
Modo 10	0,0037	0,00365	0,00%
Modo 11	0,0037	0,003649	0,03%
Modo 12	0,0032	0,003222	-0,06%
Modo 13	0,0032	0,003221	-0,03%
Modo 14	0,0030	0,00298	0,00%
Modo 15	0,0027	0,002701	-0,04%
Modo 16	0,0027	0,002701	-0,04%

Tabla 9, Periodos modelo 3.

	Formar modales normalizadas						
	SAFE_TB						
	Modo 1	Modo 2	Modo 3	Modo 4	Modo 5	Modo 6	Modo 7
GDL 1	1	1	1	0	0	0	0
GDL 2	0	0	0	0	1	1	1
GDL 3	1	1	1	0	0	0	0
GDL 4	0	0	0	0	1	-1	1
GDL 5	0	-1	1	-1	0	0	0
GDL 6	0	0	0	0	0	0	-1
GDL 7	0	-1	1	-1	0	0	0
GDL 8	0	0	0	0	0	0	-1
GDL 9	1	-1	0	1	0	0	0
GDL 10	0	0	0	0	1	1	-1
GDL 11	1	-1	0	1	0	0	0
GDL 12	0	0	0	0	1	-1	-1
GDL 13	1	0	-1	-1	0	0	0
GDL 14	0	0	0	0	1	1	0
GDL 15	1	0	-1	-1	0	0	0
GDL 16	0	0	0	0	1	-1	0

	SAP2000						
	Modo 1	Modo 2	Modo 3	Modo 4	Modo 5	Modo 6	Modo 7
GDL 1	1	1	1	0	0	0	0
GDL 2	0	0	0	0	1	1	1
GDL 3	1	1	1	0	0	0	0
GDL 4	0	0	0	0	1	-1	1
GDL 5	0	-1	1	-1	0	0	0
GDL 6	0	0	0	0	0	0	-1
GDL 7	0	-1	1	-1	0	0	0
GDL 8	0	0	0	0	0	0	-1
GDL 9	1	-1	0	1	0	0	0
GDL 10	0	0	0	0	1	1	-1
GDL 11	1	-1	0	1	0	0	0
GDL 12	0	0	0	0	1	-1	-1
GDL 13	1	0	-1	-1	0	0	0
GDL 14	0	0	0	0	1	1	0
GDL 15	1	0	-1	-1	0	0	0
GDL 16	0	0	0	0	1	-1	0

	ξ						
	Modo 1	Modo 2	Modo 3	Modo 4	Modo 5	Modo 6	Modo 7
GDL 1	0,00%	0,00%	0,00%	0,00%	0,00%	0,42%	0,00%
GDL 2	0,00%	-0,05%	0,00%	0,00%	0,00%	0,00%	0,02%
GDL 3	0,00%	0,00%	0,00%	0,00%	0,00%	0,42%	0,00%
GDL 4	0,00%	-0,05%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 5	0,01%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 6	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,03%
GDL 7	0,01%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 8	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 9	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 10	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,03%
GDL 11	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 12	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 13	0,00%	0,07%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 14	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,03%
GDL 15	0,00%	0,07%	0,00%	0,00%	0,00%	0,00%	0,00%
GDL 16	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Tabla 10, modos de vibrar normalizados modelo 3.

	Desplazamientos por nodo Análisis Modal Espectral		
	SAFE_TB		
	Ux	Uy	θy
Nodo 1	0	0	0
Nodo 2	0	0	0
Nodo 3	0,0135	0	0,0039
Nodo 4	0,0135	0	0,0039
Nodo 5	0,0303	0	0,0035
Nodo 6	0,0303	0	0,0035
Nodo 7	0,0430	0	0,0025
Nodo 8	0,0430	0	0,0025
Nodo 9	0,0498	0	0,0013
Nodo 10	0,0498	0	0,0013

	SAP2000		
	Ux	Uy	θy
Nodo 1	0	0	0
Nodo 2	0	0	0
Nodo 3	0,0135	0	0,0039
Nodo 4	0,0135	0	0,0039
Nodo 5	0,0304	0	0,0035
Nodo 6	0,0304	0	0,0035
Nodo 7	0,0430	0	0,0025
Nodo 8	0,0430	0	0,0025
Nodo 9	0,0498	0	0,0013
Nodo 10	0,0498	0	0,0013

	ξ		
	Ux	Uy	θy
Nodo 1	0,00%	0,00%	0,00%
Nodo 2	0,00%	0,00%	0,00%
Nodo 3	-0,04%	0,00%	-0,05%
Nodo 4	-0,04%	0,00%	-0,05%
Nodo 5	-0,05%	0,00%	-0,03%
Nodo 6	-0,05%	0,00%	-0,03%
Nodo 7	-0,05%	0,00%	-0,04%
Nodo 8	-0,05%	0,00%	-0,04%
Nodo 9	-0,05%	0,00%	-0,08%
Nodo 10	-0,05%	0,00%	-0,08%

Tabla 11, desplazamientos análisis modal espectral modelo 3.

		Fuerzas resistentes Análisis Modal Espectral SAFE_TB		
Elemento		Fx	Fz	My
1	Start	0,2896	0,0721	0,1341
1	End	0,2896	0,0721	0,0826
2	Start	0,2896	0,0721	0,1341
2	End	0,2896	0,0721	0,0826
3	Start	0	0,1072	0,1607
3	End	0	0,1072	0,1607
4	Start	0,1934	0,0601	0,0884
4	End	0,1934	0,0601	0,0929
5	Start	0,1934	0,0601	0,0884
5	End	0,1934	0,0601	0,0929
6	Start	0	0,0977	0,1465
6	End	0	0,0977	0,1465
7	Start	0,1028	0,0476	0,0659
7	End	0,1028	0,0476	0,0781
8	Start	0,1028	0,0476	0,0659
8	End	0,1028	0,0476	0,0781
9	Start	0	0,0695	0,1042
9	End	0	0,0695	0,1042
10	Start	0,0348	0,0301	0,0393
10	End	0,0348	0,0301	0,0522
11	Start	0,0348	0,0301	0,0393
11	End	0,0348	0,0301	0,0522
12	Start	0	0,0348	0,0522
12	End	0	0,0348	0,0522

		SAP2000		
Elemento		Fx	Fz	My
1	Start	0,2895	0,0720	0,1341
1	End	0,2895	0,0720	0,0825
2	Start	0,2895	0,0720	0,1341
2	End	0,2895	0,0720	0,0825
3	Start	1,09E-12	0,1071	0,1607
3	End	1,09E-12	0,1071	0,1607
4	Start	0,1933	0,0601	0,0883
4	End	0,1933	0,0601	0,0929
5	Start	0,1933	0,0601	0,0883
5	End	0,1933	0,0601	0,0929
6	Start	3,38E-12	0,0977	0,1465
6	End	3,38E-12	0,0977	0,1465
7	Start	0,1027	0,0476	0,0659
7	End	0,1027	0,0476	0,0781
8	Start	0,1027	0,0476	0,0659
8	End	0,1027	0,0476	0,0781
9	Start	9,31E-13	0,0695	0,1042
9	End	9,31E-13	0,0695	0,1042
10	Start	0,0348	0,0301	0,0392
10	End	0,0348	0,0301	0,0522
11	Start	0,0348	0,0301	0,0392
11	End	0,0348	0,0301	0,0522
12	Start	1,15E-12	0,0348	0,0522
12	End	1,15E-12	0,0348	0,0522

		ξ		
Elemento		Fx	Fz	My
1	Start	0,03%	0,14%	0,01%
1	End	0,03%	0,14%	0,11%
2	Start	0,03%	0,14%	0,01%
2	End	0,03%	0,14%	0,11%
3	Start	0,00%	0,09%	0,02%
3	End	0,00%	0,09%	0,02%
4	Start	0,05%	0,00%	0,08%
4	End	0,05%	0,00%	0,02%
5	Start	0,05%	0,00%	0,08%
5	End	0,05%	0,00%	0,02%
6	Start	0,00%	0,00%	0,01%
6	End	0,00%	0,00%	0,01%
7	Start	0,10%	0,00%	0,05%
7	End	0,10%	0,00%	0,00%
8	Start	0,10%	0,00%	0,05%
8	End	0,10%	0,00%	0,00%
9	Start	0,00%	0,00%	0,02%
9	End	0,00%	0,00%	0,02%
10	Start	0,00%	0,00%	0,15%
10	End	0,00%	0,00%	0,00%
11	Start	0,00%	0,00%	0,15%
11	End	0,00%	0,00%	0,00%
12	Start	0,00%	0,00%	0,00%
12	End	0,00%	0,00%	0,00%

Tabla 12, fuerzas resistentes análisis modal espectral.

5.3. Estructuras en 3D

5.3.1. Marco 3D de un piso

Número de grados de libertad: 24

Número de elementos: 8

Tipo de elementos: ElasticBeamColumn3D

Apoyos: Empotrados

Diafragmas rígidos: No

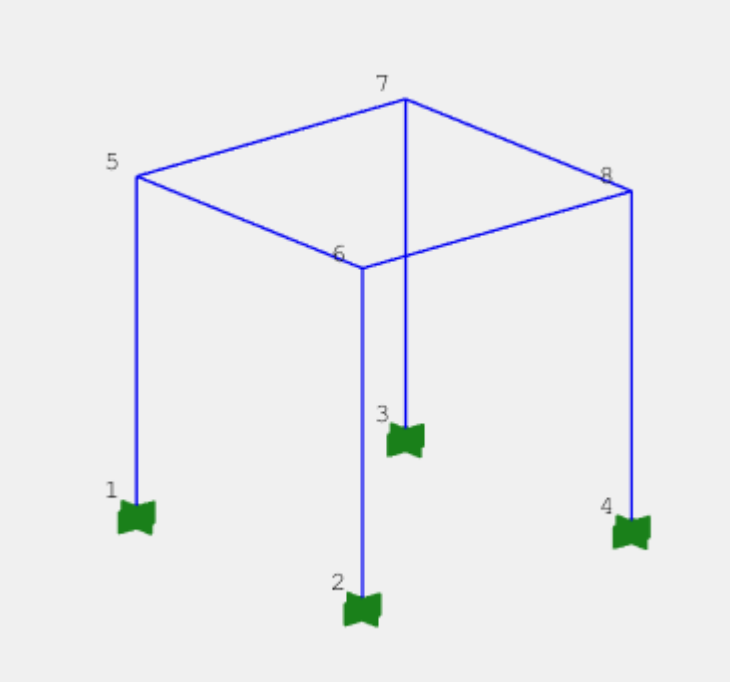

Figura 11, modelo 4, marco 3D.

Para comenzar el análisis de estructuras en 3D, se analiza un marco en dos direcciones con masas puntuales de 0,1 [ton] las cuales son incorporadas en los nodos superiores de la estructura. Por otro lado, a los nodos inferiores se les restringen los 6 grados de libertad. La estructura está compuesta por 8 elementos viga columna, estos son todos iguales con excepción del elemento contenido entre los nodos ‘1’ y ‘5’. Este último tiene un módulo de elasticidad de 40.000.000 [tonf/m^2], de esta manera se genera un desplazamiento del centro de rigidez con respecto al centro de masa, lo que produce una torsión en el sentido perpendicular al plano. Los demás elementos poseen un módulo de elasticidad de 20.000.000 [tonf/m^2].

Los grados de libertad del sistema corresponden a los desplazamientos y giros en los tres planos para los nodos 5, 6, 7 y 8.

Los códigos de la implementación de este modelo para SAFE_TB, se encuentran en el Anexo 4.

Resultados:

	SAFE_TB	SAP2000	ξ
Modo 1	1,0597	1,060055	-0,04%
Modo 2	1,0313	1,031651	-0,04%
Modo3	0,8934	0,893658	-0,03%
Modo 4	0,5454	0,545621	-0,04%
Modo 5	0,0154	0,01539	0,00%
Modo 6	0,0154	0,015389	0,00%
Modo 7	0,0154	0,015387	0,00%
Modo 8	0,0109	0,010882	0,00%
Modo 9	0,0109	0,010882	0,00%
Modo 10	0,0109	0,010882	0,00%
Modo 11	0,0109	0,010882	0,00%
Modo 12	0,0109	0,010882	0,00%

Tabla 13, periodos modelo 4.

	Formar modales normalizadas			
	SAFE_TB			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	-0,3971	0,8964	-0,9999	0,9999
GDL 2	0,3971	0,8964	0,9999	0,9999
GDL 3	0,0000	0,0002	0,0000	0,0002
GDL 4	-0,3971	0,8965	-1,0000	1,0000
GDL 5	1,0000	1,0000	-0,3971	-0,8964
GDL 6	0,0002	0,0000	0,0001	-0,0003
GDL 7	-1,0000	1,0000	0,3971	-0,8964
GDL 8	0,3971	0,8965	1,0000	1,0000
GDL 9	-0,0002	0,0000	-0,0001	-0,0003
GDL 10	-1,0000	1,0000	0,3971	-0,8964
GDL 11	1,0000	1,0000	-0,3971	-0,8964
GDL 12	0,0000	-0,0003	0,0000	0,0002

	SAP2000			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	-0,3972	0,8964	-0,9999	0,9999
GDL 2	0,3972	0,8964	0,9999	0,9999
GDL 3	0,0000	0,0002	0,0000	0,0002
GDL 4	-0,3973	0,8965	-1,0000	1,0000
GDL 5	1,0000	1,0000	-0,3972	-0,8964
GDL 6	0,0002	0,0000	0,0001	-0,0003
GDL 7	-1,0000	1,0000	0,3972	-0,8964
GDL 8	0,3973	0,8965	1,0000	1,0000
GDL 9	-0,0002	0,0000	-0,0001	-0,0003
GDL 10	-1,0000	1,0000	0,3973	-0,8964
GDL 11	1,0000	1,0000	-0,3973	-0,8964
GDL 12	0,0000	-0,0003	0,0000	0,0002

	ξ			
	Modo 1	Modo 2	Modo 3	Modo 4
GDL 1	-0,03%	0,00%	0,00%	0,00%
GDL 2	-0,03%	0,00%	0,00%	0,00%
GDL 3	0,00%	0,00%	0,00%	0,00%
GDL 4	-0,03%	0,00%	0,00%	0,00%
GDL 5	0,00%	0,00%	-0,03%	0,00%
GDL 6	0,00%	0,00%	0,00%	0,00%
GDL 7	0,00%	0,00%	-0,03%	0,00%
GDL 8	-0,03%	0,00%	0,00%	0,00%
GDL 9	0,00%	0,00%	0,00%	0,00%
GDL 10	0,00%	0,00%	-0,04%	0,00%
GDL 11	0,00%	0,00%	-0,04%	0,00%
GDL 12	0,00%	0,00%	0,00%	0,00%

Tabla 14, modos de vibrar normalizados modelo 4.

	Desplazamientos por nodo Análisis Modal Espectral					
	SAFE_TB					
	Ux	Uy	Uz	θx	θy	θz
Nodo 1	0	0	0	0	0	0
Nodo 2	0	0	0	0	0	0
Nodo 3	0	0	0	0	0	0
Nodo 4	0	0	0	0	0	0
Nodo 5	0,0251	0,0086	0	0,0025	0,0076	0,0037
Nodo 6	0,0251	0,0079	0	0,0015	0,0046	0,0042
Nodo 7	0,0344	0,0086	0	0,0014	0,0070	0,0042
Nodo 8	0,0344	0,0079	0	0,0015	0,0066	0,0041

	SAP2000					
	Ux	Uy	Uz	θx	θy	θz
Nodo 1	0	0	0	0	0	0
Nodo 2	0	0	0	0	0	0
Nodo 3	0	0	0	0	0	0
Nodo 4	0	0	0	0	0	0
Nodo 5	0,0251	0,0086	0	0,0025	0,0076	0,0037
Nodo 6	0,0252	0,0079	0	0,0015	0,0046	0,0042
Nodo 7	0,0345	0,0086	0	0,0014	0,0070	0,0042
Nodo 8	0,0345	0,0079	0	0,0015	0,0066	0,0041

	ξ					
	Ux	Uy	θy	θx	θy	θz
Nodo 1	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 2	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 3	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 4	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 5	-0,04%	-0,02%	0,00%	0,00%	-0,03%	-0,03%
Nodo 6	-0,04%	-0,01%	0,00%	0,00%	-0,04%	-0,02%
Nodo 7	-0,04%	-0,03%	0,00%	-0,07%	-0,03%	-0,02%
Nodo 8	-0,04%	-0,01%	0,00%	0,00%	-0,03%	0,00%

Tabla 15, desplazamientos análisis modal espectral modelo 4.

		Desplazamientos por nodo Análisis Modal Espectral					
		SAFE_TB					
		Ux	Uy	Uz	θx	θy	θz
1	Start	0,1279	0,0456	0,0978	0,0856	0,2443	0,0218
1	End	0,1279	0,0456	0,0978	0,0512	0,1395	0,0218
2	Start	0,0846	0,0267	0,0876	0,0450	0,1428	0,0124
2	End	0,0846	0,0267	0,0876	0,0351	0,1111	0,0124
3	Start	0,1111	0,0304	0,0819	0,0504	0,1908	0,0124
3	End	0,1111	0,0304	0,0819	0,0408	0,1424	0,0124
4	Start	0,1134	0,0265	0,0917	0,0449	0,1932	0,0119
4	End	0,1134	0,0265	0,0917	0,0347	0,1471	0,0119
5	Start	0,0212	0,0077	0,0842	0,0082	0,1366	0,0120
5	End	0,0212	0,0077	0,0842	0,0082	0,1159	0,0113
6	Start	0,0131	0,0086	0,0269	0,0441	0,0070	0,0211
6	End	0,0131	0,0086	0,0269	0,0366	0,0070	0,0182
7	Start	0,0112	0,0016	0,0203	0,0304	0,0079	0,0164
7	End	0,0112	0,0016	0,0203	0,0305	0,0079	0,0171
8	Start	0,0014	0,0076	0,0944	0,0058	0,1428	0,0114
8	End	0,0014	0,0076	0,0944	0,0058	0,1404	0,0113

		SAP2000					
		Ux	Uy	Uz	θx	θy	θz
1	Start	0,1279	0,0456	0,0977	0,0856	0,2443	0,0218
1	End	0,1279	0,0456	0,0977	0,0512	0,1394	0,0218
2	Start	0,0846	0,0267	0,0875	0,0450	0,1428	0,0124
2	End	0,0846	0,0267	0,0875	0,0351	0,1111	0,0124
3	Start	0,1110	0,0304	0,0818	0,0504	0,1908	0,0124
3	End	0,1110	0,0304	0,0818	0,0408	0,1424	0,0124
4	Start	0,1134	0,0265	0,0916	0,0449	0,1931	0,0120
4	End	0,1134	0,0265	0,0916	0,0347	0,1471	0,0120
5	Start	0,0212	0,0077	0,0841	0,0082	0,1365	0,0120
5	End	0,0212	0,0077	0,0841	0,0082	0,1159	0,0113
6	Start	0,0131	0,0086	0,0269	0,0441	0,0070	0,0211
6	End	0,0131	0,0086	0,0269	0,0365	0,0070	0,0182
7	Start	0,0112	0,0016	0,0203	0,0304	0,0079	0,0164
7	End	0,0112	0,0016	0,0203	0,0304	0,0079	0,0171
8	Start	0,0014	0,0076	0,0944	0,0058	0,1427	0,0114
8	End	0,0014	0,0076	0,0944	0,0058	0,1403	0,0113

	ξ					
	Ux	Uy	θy	θx	θy	θz
1 Start	0,01%	0,01%	0,07%	0,03%	0,02%	-0,03%
1 End	0,01%	0,01%	0,07%	0,05%	0,04%	-0,03%
2 Start	0,06%	0,03%	0,08%	0,05%	0,03%	-0,01%
2 End	0,06%	0,03%	0,08%	0,07%	0,06%	-0,01%
3 Start	0,07%	0,00%	0,06%	0,05%	0,03%	-0,01%
3 End	0,07%	0,00%	0,06%	0,06%	0,05%	-0,01%
4 Start	0,02%	0,12%	0,06%	0,05%	0,03%	-0,01%
4 End	0,02%	0,12%	0,06%	0,07%	0,04%	-0,01%
5 Start	-0,12%	-0,51%	0,09%	0,01%	0,05%	-0,02%
5 End	-0,12%	-0,51%	0,09%	0,01%	0,06%	-0,01%
6 Start	-0,11%	0,28%	-0,04%	0,06%	0,01%	-0,01%
6 End	-0,11%	0,28%	-0,04%	0,06%	0,01%	-0,01%
7 Start	-0,41%	0,00%	-0,08%	0,10%	-0,01%	-0,01%
7 End	-0,41%	0,00%	-0,08%	0,07%	-0,01%	0,01%
8 Start	-0,43%	-0,36%	0,00%	-0,05%	0,05%	0,04%
8 End	-0,43%	-0,36%	0,00%	-0,05%	0,05%	0,04%

Tabla 16, fuerzas resistentes modelo 4.

5.3.3. Marco 3D en dos direcciones de un piso con diafragma rígido

Número de grados de libertad: 15

Número de elementos: 8

Tipo de elementos: ElasticBeamColumn3D

Apoyos: Empotrados

Diafragmas rígidos: No

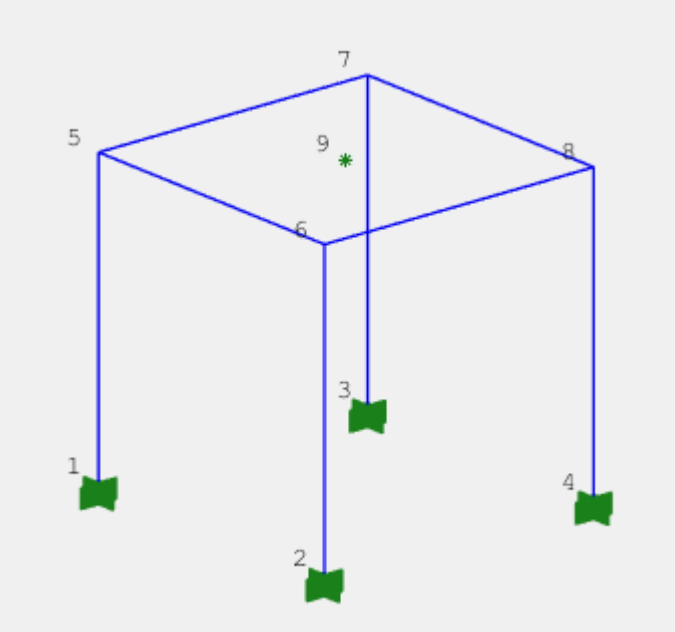

Figura 12, marco en 3D con diafragma rígido.

Este modelo es igual al anterior con la diferencia que este incorpora un diafragma rígido entre todos los puntos del nivel superior. Esto se refleja en la integración de 4 restricciones de tipo diafragma entre los nodos 5, 6, 7 y 8 asociados al nodo 9.

Los grados de libertad del sistema corresponden al desplazamiento en la vertical y los giros en los ejes horizontales para los nodos 5, 6, 7 y 8 y los desplazamientos horizontales más el giro en el eje vertical para el nodo 9.

Los códigos de la implementación de este modelo para SAFE_TB, se encuentran en el Anexo 5.

Resultados:

	Periodo [seg]		
	SAFE_TB	SAP2000	ξ
Modo 1	1,0579	1,0583	-0,04%
Modo 2	1,0273	1,0277	-0,04%
Modo3	0,8875	0,8878	-0,03%
Modo 4	0,0154	0,0154	0,00%
Modo 5	0,0154	0,0154	0,00%
Modo 6	0,0154	0,0154	0,00%
Modo 7	0,0109	0,0109	0,00%

Tabla 17, Periodos modelo 5.

	Formar modales normalizadas		
	SAFE_TB		
	Modo 1	Modo 2	Modo 3
GDL 1	0,0000	0,0002	0,0000
GDL 2	-0,0004	0,0000	-0,0003
GDL 3	0,0004	0,0000	0,0003
GDL 4	0,0000	-0,0003	0,0000
GDL 5	1,0000	1,0000	0,6138
GDL 6	-1,0000	1,0000	-0,6138
GDL 7	-0,2728	0,0000	1,0000

	SAP2000		
	Modo 1	Modo 2	Modo 3
GDL 1	0,0000	0,0002	0,0000
GDL 2	-0,0004	0,0000	-0,0003
GDL 3	0,0004	0,0000	0,0003
GDL 4	0,0000	-0,0003	0,0000
GDL 5	1,0000	1,0000	0,6135
GDL 6	-1,0000	1,0000	-0,6135
GDL 7	-0,2727	0,0000	1,0000
	ξ		
	Modo 1	Modo 2	Modo 3
GDL 1	0,00%	0,00%	0,00%
GDL 2	0,00%	0,00%	0,00%
GDL 3	0,00%	0,00%	0,00%
GDL 4	0,00%	0,00%	0,00%
GDL 5	0,00%	0,00%	0,04%
GDL 6	0,00%	0,00%	0,04%
GDL 7	0,04%	0,00%	0,00%

Tabla 18, modos de vibrar normalizados modelo 5.

	Desplazamientos por nodo Análisis Modal Espectral					
	SAFE_TB					
	Ux	Uy	Uz	θx	θy	θz
Nodo 1	0	0	0	0	0	0
Nodo 2	0	0	0	0	0	0
Nodo 3	0	0	0	0	0	0
Nodo 4	0	0	0	0	0	0
Nodo 5	0,0260	0,0089	0	0,0026	0,0078	0,0043
Nodo 6	0,0260	0,0084	0	0,0016	0,0047	0,0043
Nodo 7	0,0335	0,0089	0	0,0014	0,0068	0,0043
Nodo 8	0,0335	0,0084	0	0,0016	0,0065	0,0043
Nodo 9	0,0293	0,0059	0	0	0	0,0043

	SAP2000					
	Ux	Uy	Uz	θx	θy	θz
Nodo 1	0	0	0	0	0	0
Nodo 2	0	0	0	0	0	0
Nodo 3	0	0	0	0	0	0
Nodo 4	0	0	0	0	0	0
Nodo 5	0,0260	0,0090	3,08E-06	0,0026	0,0078	0,0043
Nodo 6	0,0260	0,0084	5,53E-06	0,0016	0,0047	0,0043
Nodo 7	0,0335	0,0090	4,7E-06	0,0014	0,0068	0,0043
Nodo 8	0,0335	0,0084	5,25E-06	0,0016	0,0065	0,0043
Nodo 9	0,0293	0,0059	0	0	0	0,0043

	ξ					
	Ux	Uy	Uz	θx	θy	θz
Nodo 1	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 2	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 3	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 4	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Nodo 5	-0,04%	-0,03%	0,00%	-0,04%	-0,03%	0,00%
Nodo 6	-0,04%	-0,01%	0,00%	0,00%	-0,04%	0,00%
Nodo 7	-0,04%	-0,03%	0,00%	0,00%	-0,03%	0,00%
Nodo 8	-0,04%	-0,01%	0,00%	0,00%	-0,03%	0,00%
Nodo 9	-0,04%	-0,03%	0,00%	0,00%	0,00%	0,00%

Tabla 19, desplazamientos análisis modal espectral modelo 5.

		Fuerzas resistentes Análisis Modal Espectral					
		SAFE_TB					
		Fx	Fy	Fz	Mx	My	Mz
1	Start	0,1324	0,0472	0,1026	0,0886	0,2527	0,0250
1	End	0,1324	0,0472	0,1026	0,0530	0,1446	0,0250
2	Start	0,0877	0,0284	0,0922	0,0479	0,1478	0,0125
2	End	0,0877	0,0284	0,0922	0,0373	0,1152	0,0125
3	Start	0,1078	0,0314	0,0784	0,0522	0,1854	0,0125
3	End	0,1078	0,0314	0,0784	0,0422	0,1381	0,0125
4	Start	0,1103	0,0282	0,0876	0,0477	0,1879	0,0125
4	End	0,1103	0,0282	0,0876	0,0369	0,1430	0,0125
5	Start	0	0	0,0867	0,0084	0,1408	0
5	End	0	0	0,0867	0,0084	0,1193	0
6	Start	0	0	0,0279	0,0457	0,0071	0
6	End	0	0	0,0279	0,0379	0,0071	0
7	Start	0	0	0,0216	0,0325	0,0071	0
7	End	0	0	0,0216	0,0325	0,0071	0
8	Start	0	0	0,0922	0,0060	0,1396	0
8	End	0	0	0,0922	0,0060	0,1371	0

		SAP2000					
		F1	F2	F3	M1	M2	M3
1	Start	0,1324	0,0472	0,1025	0,0886	0,2527	0,0250
1	End	0,1324	0,0472	0,1025	0,0529	0,1445	0,0250
2	Start	0,0876	0,0284	0,0922	0,0479	0,1478	0,0125
2	End	0,0876	0,0284	0,0922	0,0373	0,1151	0,0125
3	Start	0,1078	0,0314	0,0784	0,0521	0,1853	0,0125
3	End	0,1078	0,0314	0,0784	0,0422	0,1380	0,0125
4	Start	0,1102	0,0282	0,0876	0,0477	0,1878	0,0125
4	End	0,1102	0,0282	0,0876	0,0369	0,1429	0,0125
5	Start	0	4,84E-18	0,0867	0,0084	0,1407	1,37E-17
5	End	0	4,84E-18	0,0867	0,0084	0,1193	2,46E-17
6	Start	5,54E-18	0	0,0279	0,0457	0,0071	7,1E-18
6	End	5,54E-18	0	0,0279	0,0379	0,0071	9,54E-18
7	Start	5,54E-18	0	0,0216	0,0324	0,0071	9,54E-18
7	End	5,54E-18	0	0,0216	0,0325	0,0071	7,1E-18
8	Start	4,84E-18	0	0,0922	0,0060	0,1396	1,37E-17
8	End	4,84E-18	0	0,0922	0,0060	0,1371	2,46E-17

		ξ					
		F1	F2	F3	M1	M2	M3
1	Start	0,03%	-0,03%	0,05%	0,03%	0,02%	-0,01%
1	End	0,03%	-0,03%	0,05%	0,05%	0,05%	-0,01%
2	Start	0,08%	-0,09%	0,04%	0,06%	0,03%	-0,05%
2	End	0,08%	-0,09%	0,04%	0,07%	0,05%	-0,05%
3	Start	0,01%	0,16%	0,04%	0,05%	0,03%	-0,05%
3	End	0,01%	0,16%	0,04%	0,07%	0,05%	-0,05%
4	Start	0,06%	0,06%	0,01%	0,06%	0,03%	-0,05%
4	End	0,06%	0,06%	0,01%	0,06%	0,05%	-0,05%
5	Start	0,00%	0,00%	-0,01%	-0,04%	0,05%	0,00%
5	End	0,00%	0,00%	-0,01%	-0,04%	0,05%	0,00%
6	Start	0,00%	0,00%	-0,11%	0,06%	0,00%	0,00%
6	End	0,00%	0,00%	-0,11%	0,07%	0,00%	0,00%
7	Start	0,00%	0,00%	0,21%	0,09%	0,04%	0,00%
7	End	0,00%	0,00%	0,21%	0,07%	0,04%	0,00%
8	Start	0,00%	0,00%	0,05%	-0,05%	0,05%	0,00%
8	End	0,00%	0,00%	0,05%	-0,05%	0,05%	0,00%

Tabla 20, fuerzas resistentes análisis modal espectral modelo 5.

CAPÍTULO VI

CONCLUSIONES

En este capítulo se presentan las conclusiones, comentarios y recomendaciones sobre la implementación y uso de los tópicos desarrollados en este trabajo de título. De esta manera, se facilita el manejo y la comprensión de quien quisiese utilizar el análisis modal espectral o la restricción de diafragma rígido dentro de SAFE_TB.

6.1. Conclusiones

1. Se cumple de manera satisfactoria con el objetivo principal, el cual corresponde a la implementación del análisis modal espectral. Esto se logró mediante la implementación de las clases ‘eigenAnalysis’ y ‘modalSystemOfEquations’, las cuales contienen funciones de uso exclusivo de este método de análisis. Estas clases son compatibles con el resto de la plataforma e interactúan con clases implementadas anteriormente.
2. Se cumple con el objetivo secundario de implementar la opción de asignar diafragmas rígidos en SAFE_TB. Esta restricción se integra en el modelo mediante la clase ‘rigidDiaphragm_Constraint’, la cual es una sub clase de ‘MP_Constraint’. Si bien, son muchas las propiedades y métodos que se heredan de la clase madre; se incorporaron los métodos y propiedades necesarias para obtener la relación que se generan entre los grados de libertad que pertenecen al diafragma.
3. Si se observan los primeros resultados, donde se presentan los periodos y los modos principales de vibrar de cada una de las estructuras, se puede ver que el error no sobrepasa el 0,1%. Este error se considera admisible; ya que proviene de algunas diferencias internas entre SAFE_TB y SAP2000, al momento de definir los parámetros iniciales. El porcentaje de error aumenta al comparar los desplazamientos y fuerzas resistentes, pero se mantienen por debajo del 1%. Esto se debe a que estos resultados se determinan a partir de parámetros con error, lo que provoca una propagación de estos; y en el caso de aplicar CQC se pierde la linealidad que se podría esperar del error.
4. Se genera una serie de recomendaciones de uso, tanto para el análisis modal espectral como para las restricciones de tipo diafragma rígido.

A continuación, se presenta recomendaciones para el uso de las clases implementadas en este trabajo.

1. En caso que se desee incorporar un nuevo elemento a la plataforma o se desee utilizar alguno que no haya sido probado en este trabajo de título, se le debe incorporar la propiedad 'modalResistingForce' y las funciones 'setModalResistingForce' y 'getModalResistingForce' como se explica en…
2. Si se incluye en el modelo algún elemento 'rigidDiaphragm_Constraint', se debe incluir también una restricción simple; 'SP_Constraint', asociada al nodo maestro del diafragma. En esta se restringen los grados de libertad que no estén incluidos en el 'MP_C', los que corresponden a; el desplazamiento en el sentido ortogonal y los giros en el plano del diafragma. De lo contrario los grados de libertad no quedan asociados a modelo y la plataforma arroja error.
3. La información almacenada en la propiedad 'modalResistingForce', se encuentra ordenada según los ejes globales del sistema. Esto se debe tener en consideración si los ejes locales del elemento se encuentran rotados.
4. La información almacenada en la propiedad 'modalResistingForce' en las clases correspondientes a los elementos, se encuentra ordenada según los ejes globales y a los nodos que contienen el elemento. De esta manera se almacena un vector con; fuerza en 'x', fuerza en 'y', fuerza en 'z', momento en 'x', momento en 'y' y momento en 'z', para el nodo final y luego para el nodo inicial.
5. Para determinar las fuerzas resistentes en un punto intermedio del elemento, este se debe dividir cuando se define el modelo. De esta manera, el valor deseado corresponde a las fuerzas en el nodo

BIBLIOGRAFÍA

- Rojas, F. (2015) CI4203: Dinámica de estructuras. Santiago, Chile.
- Boroschek, R; Soto, P; Leon, R (2010) Informe red local de registros de edificios ‘Cámara chilena de la construcción’. Santiago, edificio ‘Cámara chilena de la construcción’, febrero 27, 2010 hora 3:34. Santiago, Chile.
- Hurtado, J. ( ) Análisis matricial de estructuras. Bogotá, Colombia.
- Rojas, F. ( ) CI5211: Análisis matricial de estructuras. Santiago, Chile
- Rojas, F. (2012) Development of a nonlinear quadrilateral layered membrane element with drilling degrees of freedom and a nonlinear quadrilateral thin flat layered shell element for the modeling of reinforced concrete walls. California, USA.
- NCh 433. Of 1996. Mod 2012. Diseño sísmico de edificios.

ANEXOS

Anexo 1

```
clc
clear all;
tic
% We proceed to define the TrussSection Object in 1D

%% Create the Model
modelObj=Model();

%% Create the Nodes: Node(tag,numberDOf,coord)
numDOF=3;
nodesObject_1 = Node(1,numDOF,[0 0]);
nodesObject_2 = Node(2,numDOF,[3 0]);
nodesObject_3 = Node(3,numDOF,[0 3]);
nodesObject_4 = Node(4,numDOF,[3 3]);
% Add the Nodes that define the model
modelObj.addNode(nodesObject_1);
modelObj.addNode(nodesObject_2);
modelObj.addNode(nodesObject_3);
modelObj.addNode(nodesObject_4);
% Add mass
nodesObject_3.setMass([0.1 0 0 ; 0 0.1 0 ; 0 0 0]);
nodesObject_4.setMass([0.1 0 0 ; 0 0.1 0 ; 0 0 0]);

%% Create Element

% Material Properties
E = 20000000;

% Elastic Material With
% Create the Uniaxial Material : ElasticUniaxialMaterial(tag,E,nu,rho)
uniaxialMaterialObject = ElasticUniaxialMaterial('ElasticUMat',E);

% Create the Rectangular Profile : RectangularProfile(tag,b,h);
profileObject = RectangularProfile('RecProf',0.1,0.1);

% Create the beamSection :
ElasticBeamColumnSection(tag,lineElementProfileObject,uniaxialMaterialObject)
beamSectionObj =
ElasticBeamColumnSection('BeamColumnSec',profileObject,uniaxialMaterialObject)
;

% Create the coord transformation :
LineElementCoordTransformation2D(tag,classTag)
coordTransObj_1 = LinearLineElementCoordTransformation2D('coordtans_1');
coordTransObj_2 = LinearLineElementCoordTransformation2D('coordtans_2');
coordTransObj_3 = LinearLineElementCoordTransformation2D('coordtans_3');

%Truss Element:
ElasticBeamColumn2D(tag,nodeTag1,nodeTag2,beamColumnSectionObject,lineElementC
rdTransf2DObject)
beamObject_1 = ElasticBeamColumn2D(1,1,3,beamSectionObj,coordTransObj_1);
```

```
beamObject_2 = ElasticBeamColumn2D(2,2,4,beamSectionObj,coordTransObj_2);
beamObject_3 = ElasticBeamColumn2D(3,3,4,beamSectionObj,coordTransObj_3);

% Add the Elements that define the Model
modelObj.addElement(beamObject_1);
modelObj.addElement(beamObject_2);
modelObj.addElement(beamObject_3);

%% Create the Constraint : FixRestraint(tag,tagNodeWithSP_Constraint,dofFixed)
spcObject_N1 = FixRestraint('SPC_N1',1,[1 2 3]);
spcObject_N2 = FixRestraint('SPC_N4',2,[1 2 3]);
% We add the SP_Constraint
modelObj.addSP_Constraint(spcObject_N1);
modelObj.addSP_Constraint(spcObject_N2);

%% We set the modal analysis
% Create the AnalysisComponents
analysisModelObj = AnalysisModel();

% Create the System of Equation
systemOfEqObj = ModalSystemOfEq();

% Create the ConstraintHandler and DOFNumberer
???
constraintHandlerObj = TransformationConstraintHandler();
dofNumbererObj = TransformationDOFNumberer();

% Create the Spectrum
SP = load('Z1_SA.txt');

%% Analysis
eigenAnalysisObj =
EigenAnalysis(modelObj,constraintHandlerObj,dofNumbererObj,analysisModelObj,sy
stemOfEqObj,SP,1);
eigenAnalysisObj.analyze(2);


close
NameAxesModel ='Undeformed';
newGraphicScaleFactor = 10;
boundariesCoord = [-200 4200;-100 4500;0 0];
boundariesCoordDef = [-500 4500;-100 5500;0 0];
sizeSimbolSPC = 0.2;
shlModelGraphicHandlerObj = ModelGraphicHandler(modelObj);
%
shlModelGraphicHandlerObj.setModelResponseRecorder(shlModelResponseRecorderObj
);

fg=figure(1);
% axesModelHandlerObjects = cell(1,1);
% set(fg,'papersize',[11 8.5]);
axesModelHandlerObject = AxesModelHandler(NameAxesModel,fg);
shlModelGraphicHandlerObj.addAxesModelHandler(axesModelHandlerObject);
```

```
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,shlLoadPatternTag
,'Undeformed');
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,svlLoadPatternTag
,'Undeformed');

%shlModelGraphicHandlerObj.setTypeResponseToDraw(NameAxesModel,'E22','Undeform
ed');

shlModelGraphicHandlerObj.setGraphicDeformationScaleFactor(newGraphicScaleFact
or);


%Paso a dibujar:
% shlModelGraphicHandlerObj.setAnalysisStepNumber(1);
%
%
shlModelGraphicHandlerObj.drawElementsWithAverageResponse(axesModelHandlerObje
ct,true);

%Dibujamos:
shlModelGraphicHandlerObj.drawElements(axesModelHandlerObject);
%shlModelGraphicHandlerObj.drawElementTags(axesModelHandlerObject);
shlModelGraphicHandlerObj.drawNodeTags(axesModelHandlerObject);


shlModelGraphicHandlerObj.drawSP_Constraints(axesModelHandlerObject,sizeSimbol
SPC);

shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject,[.
5,0.7,2.5]);
shlModelGraphicHandlerObj.drawElementLoadsInLoadPattern(axesModelHandlerObject
,[.5,0.7,2.5]);
shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject);
shlModelGraphicHandlerObj.setElementPencilGraphicByComponent('ElementShadow');

%axesModelHandlerObject.drawAxesDirections(0.2,0.2);
set(gca,'XLimMode','auto');
```

Anexo 2

```
clc
clear all;
tic
% We proceed to define the TrussSection Object in 1D

%% Create the Model
modelObj=Model();

%% Create the Nodes: Node(tag,numberDOf,coord)
numDOF=3;
nodesObject_1 = Node(1,numDOF,[0 0]);
nodesObject_2 = Node(2,numDOF,[3 0]);
nodesObject_3 = Node(3,numDOF,[0 3]);
```

```
nodesObject_4 = Node(4,numDOF,[3 3]);
% Add the Nodes that define the model
modelObj.addNode(nodesObject_1);
modelObj.addNode(nodesObject_2);
modelObj.addNode(nodesObject_3);
modelObj.addNode(nodesObject_4);

%% Create Element

% Material Properties
E = 20000000;
nu = 0.2;
rho = 0.005;

% Elastic Material With
% Create the Uniaxial Material : ElasticUniaxialMaterial(tag,E,nu,rho)
uniaxialMaterialObject = ElasticUniaxialMaterial('ElasticUMat',E,nu,rho);

% Create the Rectangular Profile : RectangularProfile(tag,b,h);
profileObject = RectangularProfile('RecProf',0.05,0.05);

% Create the beamSection :
ElasticBeamColumnSection(tag,lineElementProfileObject,uniaxialMaterialObject)
beamSectionObj =
ElasticBeamColumnSection('BeamColumnSec',profileObject,uniaxialMaterialObject)
;

% Create the coord transformation :
LineElementCoordTransformation2D(tag,classTag)
coordTransObj_1 = LinearLineElementCoordTransformation2D('coordtans_1');
coordTransObj_2 = LinearLineElementCoordTransformation2D('coordtans_2');
coordTransObj_3 = LinearLineElementCoordTransformation2D('coordtans_3');

%Truss Element:
ElasticBeamColumn2D(tag,nodeTag1,nodeTag2,beamColumnSectionObject,lineElementC
rdTransf2DObject)
beamObject_1 = ElasticBeamColumn2D(1,1,3,beamSectionObj,coordTransObj_1);
beamObject_2 = ElasticBeamColumn2D(2,2,4,beamSectionObj,coordTransObj_2);
beamObject_3 = ElasticBeamColumn2D(3,3,4,beamSectionObj,coordTransObj_3);

% Add the Elements that define the Model
modelObj.addElement(beamObject_1);
modelObj.addElement(beamObject_2);
modelObj.addElement(beamObject_3);

%% Create the Constraint : FixRestraint(tag,tagNodeWithSP_Constraint,dofFixed)
spcObject_N1 = FixRestraint('SPC_N1',1,[1 2 3]);
spcObject_N2 = FixRestraint('SPC_N4',2,[1 2 3]);
% We add the SP_Constraint
modelObj.addSP_Constraint(spcObject_N1);
modelObj.addSP_Constraint(spcObject_N2);

%% We set the modal analysis
% Create the AnalysisComponents
analysisModelObj = AnalysisModel();
```

```
% Create the System of Equation
systemOfEqObj = ModalSystemOfEq();

% Create the ConstraintHandler and DOFNumberer
???
constraintHandlerObj = TransformationConstraintHandler();
dofNumbererObj = TransformationDOFNumberer();

% Create the Spectrum
SP = load('Z1_SA.txt');

%% Analysis
eigenAnalysisObj =
EigenAnalysis(modelObj,constraintHandlerObj,dofNumbererObj,analysisModelObj,sy
stemOfEqObj,SP,1);
eigenAnalysisObj.analyze(2);



close
NameAxesModel ='Undeformed';
newGraphicScaleFactor = 10;
boundariesCoord = [-200 4200;-100 4500;0 0];
boundariesCoordDef = [-500 4500;-100 5500;0 0];
sizeSimbolSPC = 0.2;
shlModelGraphicHandlerObj = ModelGraphicHandler(modelObj);
%
shlModelGraphicHandlerObj.setModelResponseRecorder(shlModelResponseRecorderObj
);

fg=figure(1);
% axesModelHandlerObjects = cell(1,1);
% set(fg,'papersize',[11 8.5]);
axesModelHandlerObject = AxesModelHandler(NameAxesModel,fg);
shlModelGraphicHandlerObj.addAxesModelHandler(axesModelHandlerObject);
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,shlLoadPatternTag
,'Undeformed');
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,svlLoadPatternTag
,'Undeformed');

%shlModelGraphicHandlerObj.setTypeResponseToDraw(NameAxesModel,'E22','Undeform
ed');

shlModelGraphicHandlerObj.setGraphicDeformationScaleFactor(newGraphicScaleFact
or);



%Paso a dibujar:
% shlModelGraphicHandlerObj.setAnalysisStepNumber(1);
%
%
shlModelGraphicHandlerObj.drawElementsWithAverageResponse(axesModelHandlerObje
ct,true);
```

```
%Dibujamos:
shlModelGraphicHandlerObj.drawElements(axesModelHandlerObject);
%shlModelGraphicHandlerObj.drawElementTags(axesModelHandlerObject);
shlModelGraphicHandlerObj.drawNodeTags(axesModelHandlerObject);
shlModelGraphicHandlerObj.drawElementTags(axesModelHandlerObject);


shlModelGraphicHandlerObj.drawSP_Constraints(axesModelHandlerObject,sizeSimbol
SPC);

shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject,[.
5,0.7,2.5]);
shlModelGraphicHandlerObj.drawElementLoadsInLoadPattern(axesModelHandlerObject
,[.5,0.7,2.5]);
shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject);
shlModelGraphicHandlerObj.setElementPencilGraphicByComponent('ElementShadow');

%axesModelHandlerObject.drawAxesDirections(0.2,0.2);
set(gca,'XLimMode','auto');
```

Anexo 3

```
clc
clear all;
tic
% We proceed to define the TrussSection Object in 1D

%% Create the Model
modelObj=Model();

%% Create the Nodes: Node(tag,numberDOf,coord)
numDOF=3;
nodesObject_1 = Node(1,numDOF,[0 0]);
nodesObject_2 = Node(2,numDOF,[3 0]);
nodesObject_3 = Node(3,numDOF,[0 3]);
nodesObject_4 = Node(4,numDOF,[3 3]);
nodesObject_5 = Node(5,numDOF,[0 6]);
nodesObject_6 = Node(6,numDOF,[3 6]);
nodesObject_7 = Node(7,numDOF,[0 9]);
nodesObject_8 = Node(8,numDOF,[3 9]);
nodesObject_9 = Node(9,numDOF,[0 12]);
nodesObject_10 = Node(10,numDOF,[3 12]);
% Add the Nodes that define the model
modelObj.addNode(nodesObject_1);
modelObj.addNode(nodesObject_2);
modelObj.addNode(nodesObject_3);
modelObj.addNode(nodesObject_4);
modelObj.addNode(nodesObject_5);
modelObj.addNode(nodesObject_6);
modelObj.addNode(nodesObject_7);
modelObj.addNode(nodesObject_8);
modelObj.addNode(nodesObject_9);
modelObj.addNode(nodesObject_10);

%% Create Element
```

```
% Material Properties
E = 40000000;
nu = 0.2;
rho = 0.005;

% Elastic Material With
% Create the Uniaxial Material : ElasticUniaxialMaterial(tag,E,nu,rho)
uniaxialMaterialObject = ElasticUniaxialMaterial('ElasticUMat',E,nu,rho);

% Create the Rectangular Profile : RectangularProfile(tag,b,h);
profileObject = RectangularProfile('RecProf',0.05,0.05);

% Create the beamSection :
ElasticBeamColumnSection(tag,lineElementProfileObject,uniaxialMaterialObject)
beamSectionObj =
ElasticBeamColumnSection('BeamColumnSec',profileObject,uniaxialMaterialObject)
;

% Create the coord transformation :
LineElementCoordTransformation2D(tag,classTag)
coordTransObj_1 = LinearLineElementCoordTransformation2D('coordtans_1');
coordTransObj_2 = LinearLineElementCoordTransformation2D('coordtans_2');
coordTransObj_3 = LinearLineElementCoordTransformation2D('coordtans_3');
coordTransObj_4 = LinearLineElementCoordTransformation2D('coordtans_4');
coordTransObj_5 = LinearLineElementCoordTransformation2D('coordtans_5');
coordTransObj_6 = LinearLineElementCoordTransformation2D('coordtans_6');
coordTransObj_7 = LinearLineElementCoordTransformation2D('coordtans_7');
coordTransObj_8 = LinearLineElementCoordTransformation2D('coordtans_8');
coordTransObj_9 = LinearLineElementCoordTransformation2D('coordtans_9');
coordTransObj_10 = LinearLineElementCoordTransformation2D('coordtans_10');
coordTransObj_11 = LinearLineElementCoordTransformation2D('coordtans_11');
coordTransObj_12 = LinearLineElementCoordTransformation2D('coordtans_12');

%Truss Element:
ElasticBeamColumn2D(tag,nodeTag1,nodeTag2,beamColumnSectionObject,lineElementC
rdTransf2DObject)
beamObject_1 = ElasticBeamColumn2D(1,1,3,beamSectionObj,coordTransObj_1);
beamObject_2 = ElasticBeamColumn2D(2,2,4,beamSectionObj,coordTransObj_2);
beamObject_3 = ElasticBeamColumn2D(3,3,4,beamSectionObj,coordTransObj_3);
beamObject_4 = ElasticBeamColumn2D(4,3,5,beamSectionObj,coordTransObj_4);
beamObject_5 = ElasticBeamColumn2D(5,4,6,beamSectionObj,coordTransObj_5);
beamObject_6 = ElasticBeamColumn2D(6,5,6,beamSectionObj,coordTransObj_6);
beamObject_7 = ElasticBeamColumn2D(7,5,7,beamSectionObj,coordTransObj_7);
beamObject_8 = ElasticBeamColumn2D(8,6,8,beamSectionObj,coordTransObj_8);
beamObject_9 = ElasticBeamColumn2D(9,7,8,beamSectionObj,coordTransObj_9);
beamObject_10 = ElasticBeamColumn2D(10,7,9,beamSectionObj,coordTransObj_10);
beamObject_11 = ElasticBeamColumn2D(11,8,10,beamSectionObj,coordTransObj_11);
beamObject_12 = ElasticBeamColumn2D(12,9,10,beamSectionObj,coordTransObj_12);

% Add the Elements that define the Model
modelObj.addElement(beamObject_1);
modelObj.addElement(beamObject_2);
modelObj.addElement(beamObject_3);
modelObj.addElement(beamObject_4);
```

```
modelObj.addElement(beamObject_5);
modelObj.addElement(beamObject_6);
modelObj.addElement(beamObject_7);
modelObj.addElement(beamObject_8);
modelObj.addElement(beamObject_9);
modelObj.addElement(beamObject_10);
modelObj.addElement(beamObject_11);
modelObj.addElement(beamObject_12);

%% Create the Constraint : FixRestraint(tag,tagNodeWithSP_Constraint,dofFixed)
spcObject_N1 = FixRestraint('SPC_N1',1,[1 2 3]);
spcObject_N2 = FixRestraint('SPC_N4',2,[1 2 3]);
% We add the SP_Constraint
modelObj.addSP_Constraint(spcObject_N1);
modelObj.addSP_Constraint(spcObject_N2);

%% We set the modal analysis
% Create the AnalysisComponents
analysisModelObj = AnalysisModel();

% Create the System of Equation
systemOfEqObj = ModalSystemOfEq();

% Create the ConstraintHandler and DOFNumberer
???
constraintHandlerObj = TransformationConstraintHandler();
dofNumbererObj = TransformationDOFNumberer();

% Create the Spectrum
SP = load('Z1_SA.txt');

%% Analysis
eigenAnalysisObj =
EigenAnalysis(modelObj,constraintHandlerObj,dofNumbererObj,analysisModelObj,sy
stemOfEqObj,SP,1);
eigenAnalysisObj.analyze(2);



close
NameAxesModel ='Undeformed';
newGraphicScaleFactor = 10;
boundariesCoord = [-200 4200;-100 4500;0 0];
boundariesCoordDef = [-500 4500;-100 5500;0 0];
sizeSimbolSPC = 0.2;
shlModelGraphicHandlerObj = ModelGraphicHandler(modelObj);
%
shlModelGraphicHandlerObj.setModelResponseRecorder(shlModelResponseRecorderObj
);

fg=figure(1);
% axesModelHandlerObjects = cell(1,1);
% set(fg,'papersize',[11 8.5]);
axesModelHandlerObject = AxesModelHandler(NameAxesModel,fg);
shlModelGraphicHandlerObj.addAxesModelHandler(axesModelHandlerObject);
```

```
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,shlLoadPatternTag
,'Undeformed');
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,svlLoadPatternTag
,'Undeformed');

%shlModelGraphicHandlerObj.setTypeResponseToDraw(NameAxesModel,'E22','Undeform
ed');

shlModelGraphicHandlerObj.setGraphicDeformationScaleFactor(newGraphicScaleFact
or);


%Paso a dibujar:
% shlModelGraphicHandlerObj.setAnalysisStepNumber(1);
%
%
shlModelGraphicHandlerObj.drawElementsWithAverageResponse(axesModelHandlerObje
ct,true);

%Dibujamos:
shlModelGraphicHandlerObj.drawElements(axesModelHandlerObject);
%shlModelGraphicHandlerObj.drawElementTags(axesModelHandlerObject);
shlModelGraphicHandlerObj.drawNodeTags(axesModelHandlerObject);
shlModelGraphicHandlerObj.drawElementTags(axesModelHandlerObject);


shlModelGraphicHandlerObj.drawSP_Constraints(axesModelHandlerObject,sizeSimbol
SPC);

shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject,[.
5,0.7,2.5]);
shlModelGraphicHandlerObj.drawElementLoadsInLoadPattern(axesModelHandlerObject
,[.5,0.7,2.5]);
shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject);
shlModelGraphicHandlerObj.setElementPencilGraphicByComponent('ElementShadow');

%axesModelHandlerObject.drawAxesDirections(0.2,0.2);
set(gca,'XLimMode','auto');
```

Anexo 4

```
clc
clear all;
tic
% We proceed to define the TrussSection Object in 1D

%% Create the Model
modelObj=Model();

%% Create the Nodes: Node(tag,numberDOf,coord)
numDOF=6;
nodesObject_1 = Node(1,numDOF,[ 0 0 0 ]);
nodesObject_2 = Node(2,numDOF,[ 3 0 0 ]);
```

```
nodesObject_3 = Node(3,numDOF,[ 0 3 0 ]);
nodesObject_4 = Node(4,numDOF,[ 3 3 0 ]);
nodesObject_5 = Node(5,numDOF,[ 0 0 3 ]);
nodesObject_6 = Node(6,numDOF,[ 3 0 3 ]);
nodesObject_7 = Node(7,numDOF,[ 0 3 3 ]);
nodesObject_8 = Node(8,numDOF,[ 3 3 3 ]);
% Add the Nodes that define the model
modelObj.addNode(nodesObject_1);
modelObj.addNode(nodesObject_2);
modelObj.addNode(nodesObject_3);
modelObj.addNode(nodesObject_4);
modelObj.addNode(nodesObject_5);
modelObj.addNode(nodesObject_6);
modelObj.addNode(nodesObject_7);
modelObj.addNode(nodesObject_8);
nodesObject_5.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);
nodesObject_6.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);
nodesObject_7.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);
nodesObject_8.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);

%% Create Element

% Material Properties
E_1 = 20000000;
E_2 = 40000000;

% Elastic Material With
% Create the Uniaxial Material : ElasticUniaxialMaterial(tag,E,varargin)
uniaxialMaterialObject_1 = ElasticUniaxialMaterial('ElasticUMat',E_1);
uniaxialMaterialObject_2 = ElasticUniaxialMaterial('ElasticUMat',E_2);

% Create the Rectangular Profile : RectangularProfile(tag,b,h);
profileObject = RectangularProfile('RecProf',0.05,0.05);

% Create the beamSection :
ElasticBeamColumnSection(tag,lineElementProfileObject,uniaxialMaterialObject)
beamSectionObj_1 =
ElasticBeamColumnSection('BeamColumnSec',profileObject,uniaxialMaterialObject_
1);
beamSectionObj_2 =
ElasticBeamColumnSection('BeamColumnSec',profileObject,uniaxialMaterialObject_
2);

% Create the coord transformation :
LineElementCoordTransformation2D(tag,classTag)
vecInLocalXZPlane_1 = [1 0 0];
vecInLocalXZPlane_2 = [0 1 0];
vecInLocalXZPlane_3 = [0 0 1];
coordTransObj_1 =
LinearLineElementCoordTransformation3D('LCT3D1',vecInLocalXZPlane_1);
coordTransObj_2 =
LinearLineElementCoordTransformation3D('LCT3D2',vecInLocalXZPlane_2);
```

```
coordTransObj_3 =
LinearLineElementCoordTransformation3D('LCT3D2',vecInLocalXZPlane_3);

%Truss Element:
ElasticBeamColumn2D(tag,nodeTag1,nodeTag2,beamColumnSectionObject,lineElementC
rdTransf2DObject)
beamObject_1 = ElasticBeamColumn3D(1,1,5,beamSectionObj_2,coordTransObj_2);
beamObject_2 = ElasticBeamColumn3D(2,2,6,beamSectionObj_1,coordTransObj_1);
beamObject_3 = ElasticBeamColumn3D(3,3,7,beamSectionObj_1,coordTransObj_1);
beamObject_4 = ElasticBeamColumn3D(4,4,8,beamSectionObj_1,coordTransObj_1);
beamObject_5 = ElasticBeamColumn3D(5,5,6,beamSectionObj_1,coordTransObj_3);
beamObject_6 = ElasticBeamColumn3D(6,5,7,beamSectionObj_1,coordTransObj_3);
beamObject_7 = ElasticBeamColumn3D(7,6,8,beamSectionObj_1,coordTransObj_3);
beamObject_8 = ElasticBeamColumn3D(8,7,8,beamSectionObj_1,coordTransObj_3);

% Add the Elements that define the Model
modelObj.addElement(beamObject_1);
modelObj.addElement(beamObject_2);
modelObj.addElement(beamObject_3);
modelObj.addElement(beamObject_4);
modelObj.addElement(beamObject_5);
modelObj.addElement(beamObject_6);
modelObj.addElement(beamObject_7);
modelObj.addElement(beamObject_8);

%% Create the Constraint : FixRestraint(tag,tagNodeWithSP_Constraint,dofFixed)
spcObject_N1 = FixRestraint('SPC_N1',1,[1 2 3 4 5 6 ]);
spcObject_N2 = FixRestraint('SPC_N2',2,[1 2 3 4 5 6 ]);
spcObject_N3 = FixRestraint('SPC_N3',3,[1 2 3 4 5 6 ]);
spcObject_N4 = FixRestraint('SPC_N4',4,[1 2 3 4 5 6 ]);
% We add the SP_Constraint
modelObj.addSP_Constraint(spcObject_N1);
modelObj.addSP_Constraint(spcObject_N2);
modelObj.addSP_Constraint(spcObject_N3);
modelObj.addSP_Constraint(spcObject_N4);

%% We set the modal analysis
% Create the AnalysisComponents
analysisModelObj = AnalysisModel();

% Create the System of Equation
systemOfEqObj = ModalSystemOfEq();

% Create the ConstraintHandler and DOFNumberer
???
constraintHandlerObj = TransformationConstraintHandler();
dofNumbererObj = TransformationDOFNumberer();

% Create the Spectrum
SP = load('Z1_SA.txt');

%% Analysis
eigenAnalysisObj =
EigenAnalysis(modelObj,constraintHandlerObj,dofNumbererObj,analysisModelObj,sy
stemOfEqObj,SP,1);
```

```
eigenAnalysisObj.analyze(2);


close
NameAxesModel ='Undeformed';
newGraphicScaleFactor = 10;
boundariesCoord = [-200 4200;-100 4500;0 0];
boundariesCoordDef = [-500 4500;-100 5500;0 0];
sizeSimbolSPC = 0.2;
shlModelGraphicHandlerObj = ModelGraphicHandler(modelObj);
%
shlModelGraphicHandlerObj.setModelResponseRecorder(shlModelResponseRecorderObj
);

fg=figure(1);
% axesModelHandlerObjects = cell(1,1);
% set(fg,'papersize',[11 8.5]);
axesModelHandlerObject = AxesModelHandler(NameAxesModel,fg);
shlModelGraphicHandlerObj.addAxesModelHandler(axesModelHandlerObject);
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,shlLoadPatternTag
,'Undeformed');
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,svlLoadPatternTag
,'Undeformed');

%shlModelGraphicHandlerObj.setTypeResponseToDraw(NameAxesModel,'E22','Undeform
ed');

shlModelGraphicHandlerObj.setGraphicDeformationScaleFactor(newGraphicScaleFact
or);


%Paso a dibujar:
% shlModelGraphicHandlerObj.setAnalysisStepNumber(1);
%
%
shlModelGraphicHandlerObj.drawElementsWithAverageResponse(axesModelHandlerObje
ct,true);

%Dibujamos:
shlModelGraphicHandlerObj.drawElements(axesModelHandlerObject);
%shlModelGraphicHandlerObj.drawElementTags(axesModelHandlerObject);
shlModelGraphicHandlerObj.drawNodeTags(axesModelHandlerObject);


shlModelGraphicHandlerObj.drawSP_Constraints(axesModelHandlerObject,sizeSimbol
SPC);

shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject,[.
5,0.7,2.5]);
shlModelGraphicHandlerObj.drawElementLoadsInLoadPattern(axesModelHandlerObject
,[.5,0.7,2.5]);
shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject);
shlModelGraphicHandlerObj.setElementPencilGraphicByComponent('ElementShadow');
```

```
%axesModelHandlerObject.drawAxesDirections(0.2,0.2);
set(gca,'XLimMode','auto');
```

Anexo 5

```
clc
clear all;
tic
% We proceed to define the TrussSection Object in 1D

%% Create the Model
modelObj=Model();

%% Create the Nodes: Node(tag,numberDOf,coord)
numDOF=6;
nodesObject_1 = Node(1,numDOF,[ 0 0 0 ]);
nodesObject_2 = Node(2,numDOF,[ 3 0 0 ]);
nodesObject_3 = Node(3,numDOF,[ 0 3 0 ]);
nodesObject_4 = Node(4,numDOF,[ 3 3 0 ]);
nodesObject_5 = Node(5,numDOF,[ 0 0 3 ]);
nodesObject_6 = Node(6,numDOF,[ 3 0 3 ]);
nodesObject_7 = Node(7,numDOF,[ 0 3 3 ]);
nodesObject_8 = Node(8,numDOF,[ 3 3 3 ]);
nodesObject_9 = Node(9,numDOF,[ 1.5 1.5 3 ]);
% Add the Nodes that define the model
modelObj.addNode(nodesObject_1);
modelObj.addNode(nodesObject_2);
modelObj.addNode(nodesObject_3);
modelObj.addNode(nodesObject_4);
modelObj.addNode(nodesObject_5);
modelObj.addNode(nodesObject_6);
modelObj.addNode(nodesObject_7);
modelObj.addNode(nodesObject_8);
modelObj.addNode(nodesObject_9);
nodesObject_5.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);
nodesObject_6.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);
nodesObject_7.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);
nodesObject_8.setMass([ 0.1 0 0 0 0 0 ; 0 0.1 0 0 0 0 ; 0 0 0.1 0 0 0 ; 0 0 0
0 0 0 ; 0 0 0 0 0 0 ; 0 0 0 0 0 0 ]);

%% Create Element

% Material Properties
E_1 = 20000000;
E_2 = 40000000;
nu = 0.2;

% Elastic Material With
% Create the Uniaxial Material : ElasticUniaxialMaterial(tag,E,varargin)
uniaxialMaterialObject_1 = ElasticUniaxialMaterial('ElasticUMat',E_1,nu);
uniaxialMaterialObject_2 = ElasticUniaxialMaterial('ElasticUMat',E_2,nu);
```

```
% Create the Rectangular Profile : RectangularProfile(tag,b,h);
profileObject = RectangularProfile('RecProf',0.05,0.05);

% Create the beamSection :
ElasticBeamColumnSection(tag,lineElementProfileObject,uniaxialMaterialObject)
beamSectionObj_1 =
ElasticBeamColumnSection('BeamColumnSec',profileObject,uniaxialMaterialObject_
1);
beamSectionObj_2 =
ElasticBeamColumnSection('BeamColumnSec',profileObject,uniaxialMaterialObject_
2);

% Create the coord transformation :
LineElementCoordTransformation2D(tag,classTag)
vecInLocalXZPlane_1 = [1 0 0];
vecInLocalXZPlane_2 = [0 1 0];
vecInLocalXZPlane_3 = [0 0 1];
coordTransObj_1 =
LinearLineElementCoordTransformation3D('LCT3D1',vecInLocalXZPlane_1);
coordTransObj_2 =
LinearLineElementCoordTransformation3D('LCT3D2',vecInLocalXZPlane_2);
coordTransObj_3 =
LinearLineElementCoordTransformation3D('LCT3D2',vecInLocalXZPlane_3);

%Truss Element:
ElasticBeamColumn2D(tag,nodeTag1,nodeTag2,beamColumnSectionObject,lineElementC
rdTransf2DObject)
beamObject_1 = ElasticBeamColumn3D(1,1,5,beamSectionObj_2,coordTransObj_2);
beamObject_2 = ElasticBeamColumn3D(2,2,6,beamSectionObj_1,coordTransObj_1);
beamObject_3 = ElasticBeamColumn3D(3,3,7,beamSectionObj_1,coordTransObj_1);
beamObject_4 = ElasticBeamColumn3D(4,4,8,beamSectionObj_1,coordTransObj_1);
beamObject_5 = ElasticBeamColumn3D(5,5,6,beamSectionObj_1,coordTransObj_3);
beamObject_6 = ElasticBeamColumn3D(6,5,7,beamSectionObj_1,coordTransObj_3);
beamObject_7 = ElasticBeamColumn3D(7,6,8,beamSectionObj_1,coordTransObj_3);
beamObject_8 = ElasticBeamColumn3D(8,7,8,beamSectionObj_1,coordTransObj_3);

% Add the Elements that define the Model
modelObj.addElement(beamObject_1);
modelObj.addElement(beamObject_2);
modelObj.addElement(beamObject_3);
modelObj.addElement(beamObject_4);
modelObj.addElement(beamObject_5);
modelObj.addElement(beamObject_6);
modelObj.addElement(beamObject_7);
modelObj.addElement(beamObject_8);

%% Create the Constraint : FixRestraint(tag,tagNodeWithSP_Constraint,dofFixed)
spcObject_N1 = FixRestraint('SPC_N1',1,[1 2 3 4 5 6 ]);
spcObject_N2 = FixRestraint('SPC_N2',2,[1 2 3 4 5 6 ]);
spcObject_N3 = FixRestraint('SPC_N3',3,[1 2 3 4 5 6 ]);
spcObject_N4 = FixRestraint('SPC_N4',4,[1 2 3 4 5 6 ]);
spcObject_N9 = FixRestraint('SPC_N9',9,[3 4 5]);
% We add the SP_Constraint
modelObj.addSP_Constraint(spcObject_N1);
modelObj.addSP_Constraint(spcObject_N2);
modelObj.addSP_Constraint(spcObject_N3);
```

```
modelObj.addSP_Constraint(spcObject_N4);
modelObj.addSP_Constraint(spcObject_N9);

%% Create the MP constraint
rigidDiaphragmConstraint_1 = RigidDiaphragmConstraint(1,9,5);
rigidDiaphragmConstraint_2 = RigidDiaphragmConstraint(2,9,6);
rigidDiaphragmConstraint_3 = RigidDiaphragmConstraint(3,9,7);
rigidDiaphragmConstraint_4 = RigidDiaphragmConstraint(4,9,8);
% We add the rigid diaphragm
modelObj.addMP_Constraint(rigidDiaphragmConstraint_1);
modelObj.addMP_Constraint(rigidDiaphragmConstraint_2);
modelObj.addMP_Constraint(rigidDiaphragmConstraint_3);
modelObj.addMP_Constraint(rigidDiaphragmConstraint_4);

%% We set the modal analysis
% Create the AnalysisComponents
analysisModelObj = AnalysisModel();

% Create the System of Equation
systemOfEqObj = ModalSystemOfEq();

% Create the ConstraintHandler and DOFNumberer
???
constraintHandlerObj = TransformationConstraintHandler();
dofNumbererObj = TransformationDOFNumberer();

% Create the Spectrum
SP = load('Z1_SA.txt');

%% Analysis
eigenAnalysisObj =
EigenAnalysis(modelObj,constraintHandlerObj,dofNumbererObj,analysisModelObj,sy
stemOfEqObj,SP,1);
eigenAnalysisObj.analyze(2);



close
NameAxesModel ='Undeformed';
newGraphicScaleFactor = 10;
boundariesCoord = [-200 4200;-100 4500;0 0];
boundariesCoordDef = [-500 4500;-100 5500;0 0];
sizeSimbolSPC = 0.2;
shlModelGraphicHandlerObj = ModelGraphicHandler(modelObj);
%
shlModelGraphicHandlerObj.setModelResponseRecorder(shlModelResponseRecorderObj
);

fg=figure(1);
% axesModelHandlerObjects = cell(1,1);
% set(fg,'papersize',[11 8.5]);
axesModelHandlerObject = AxesModelHandler(NameAxesModel,fg);
shlModelGraphicHandlerObj.addAxesModelHandler(axesModelHandlerObject);
```

```
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,shlLoadPatternTag
,'Undeformed');
%
shlModelGraphicHandlerObj.setLoadPatternToDraw(NameAxesModel,svlLoadPatternTag
,'Undeformed');

%shlModelGraphicHandlerObj.setTypeResponseToDraw(NameAxesModel,'E22','Undeform
ed');

shlModelGraphicHandlerObj.setGraphicDeformationScaleFactor(newGraphicScaleFact
or);


%Paso a dibujar:
% shlModelGraphicHandlerObj.setAnalysisStepNumber(1);
%
%
shlModelGraphicHandlerObj.drawElementsWithAverageResponse(axesModelHandlerObje
ct,true);

%Dibujamos:
shlModelGraphicHandlerObj.drawElements(axesModelHandlerObject);
% shlModelGraphicHandlerObj.drawElementTags(axesModelHandlerObject);
shlModelGraphicHandlerObj.drawNodeTags(axesModelHandlerObject);


shlModelGraphicHandlerObj.drawSP_Constraints(axesModelHandlerObject,sizeSimbol
SPC);

shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject,[.
5,0.7,2.5]);
shlModelGraphicHandlerObj.drawElementLoadsInLoadPattern(axesModelHandlerObject
,[.5,0.7,2.5]);
shlModelGraphicHandlerObj.drawNodeLoadsInLoadPattern(axesModelHandlerObject);
shlModelGraphicHandlerObj.setElementPencilGraphicByComponent('ElementShadow');

%axesModelHandlerObject.drawAxesDirections(0.2,0.2);
set(gca,'XLimMode','auto');
```