Desarrollo de una aplicación en entorno Matlab para la generación de curvas de dispersión en placas planas, tubos y perfiles en I de materiales isotrópicos

Enrique Ortega Pinzón

Trabajo de Grado para Optar el Título de Ingeniero Mecánico

Director
Jabid Eduardo Quiroga Méndez
PhD Ingeniería Civil

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingeniería Mecánica
Bucaramanga
2020

Dedicatoria

A mis padres, Carlos y Digneris, gracias a su esfuerzo hoy mi sueño es una realidad.

A mi novia, Erika, gracias por acompañarme en todo momento, por tu amor y apoyo constante para que hoy pueda culminar esta etapa de mi vida.

A mis hermanos y toda mi familia quienes siempre han sido mi apoyo.

¡Gracias!

Enrique Ortega Pinzón

Agradecimientos

A mi director de trabajo de grado Jabid Eduardo Quiroga Méndez, por brindarme todo su apoyo confianza y asesoramiento en la realización de este proyecto.

A la Universidad Industrial de Santander y a cada docente que hizo parte de mi formación personal y profesional. ¡Gracias!

Tabla de contenido

Introducción 10

1. Objetivos 12

1.1. Objetivo General 12

1.2. Objetivos Específicos 12

2. Ondas guiadas en NDT y SHM 13

3. Propagación de ondas en un medio sólido 15

3.1 Reflexión de ondas mecánicas 16

3.2 Ondas Lamb 18

3.2.1 Solución de modos de Lamb en placas 19

3.3 Aspectos generales de las Ondas 20

4. Extracción numérica de curvas de dispersión 21

4.1 Modelo de Elementos Finitos 22

5. Metodología 23

5.1 Post-procesamiento 25

5.1.1 Clasificación de modos armónicos 25

5.1.2 Clasificación de modos por el número de modo (p) 27

5.1.3 Clasificación de modo en A o S 28

5.1.4 Cálculo de la velocidad de fase y la velocidad de grupo 29

6. Modelo de FE para la generación de curvas de dispersión en tubos y perfiles en I _____ 30

7. Resultados _____ 33

8. Conclusiones _____ 38

9. Recomendaciones _____ 39

Referencias Bibliográficas _____ 41

Apéndices _____ 43

Lista de Figuras

Figura 1 Comparación entre el alcance de un ensayo por ultrasonido convencional 14

Figura 2 Posibles movimientos de las partículas que componen un medio infinito ebido al paso de una onda de tensión. 16

Figura 3 Reflexión de ondas en un medio sólido según la ley de Snell. 17

Figura 4 Deformación de los modos simétricos y anti-simétricos. 19

Figura 5 Modelo de FE usado para el análisis modos Lamb en placas. 24

Figura 6 Ejemplo de mallado generado con PDE toolbox en Matlab 24

Figura 7 Deformación en los bordes superior e inferior de un modo no armónico 26

Figura 8 Longitud de onda 27

Figura 9 Desplazamientos de modos simétricos y antisimétricos. 28

Figura 10 Representación método de diferencias finitas. 30

Figura 11 Acople en tubo. 31

Figura 12 Modos Lamb en tubo 32

Figura 13 Longitud de Onda Vs Frecuencia. Ejemplo 1 34

Figura 14 Número de onda Vs Frecuencia. Ejemplo 1 34

Figura 15 Velocidad de fase Vs Frecuencia. Ejemplo 1 35

Figura 16 Velocidad de Grupo Vs Frecuencia. Ejemplo 1 35

Figura 17 Longitud de Onda Vs Frecuenca. Ejemplo 2. 36

Figura 18 Número de onda Vs Frecuencia. Ejemplo 2. 36

Figura 19 Velocidad de Fase Vs Frecuencia. Ejemplo 2. 37

Figura 20 Velocidad de Grupo Vs Frecuencia. Ejemplo 2 37

Lista de Apéndices

Apéndice A: Programa principal. _____ 43

Apéndice B: Función para el análisis modal_____ 47

Apéndice C: Función para la creación de una matriz con los desplazamientos nodales. _____ 49

Apéndice D: Función para la clasificación de modos armónicos. _____ 49

Apéndice E: Función para la clasificación de modos en Simétricos (S) o Antisimétricos (A) _ 50

Apéndice F: Función para la clasificación de modos por el número de modo (p) (Longitud de onda). _____ 51

Apéndice G: Función para la clasificación de los modos antisimétricos en A0, A1, etc. Y los modos simétricos en S0, S1, etc. _____ 53

Apéndice H: Programa para el análisis modal con condiciones de contorno periódicas en 3 DOFs (para geometrías 3D)_____ 54

Apéndice I: Diseño de la interfaz gráfica de usuario (GUI) _____ 56

Resumen

Título: Desarrollo de una aplicación en entorno Matlab para la generación de curvas de dispersión de las ondas guiadas en placas planas, tubos y perfiles en I de materiales isotrópicos[*]

Autor: Enrique Ortega Pinzón[**]

Palabras Clave: Curvas de dispersión, Ondas guiadas, Elementos finitos, PDE Matlab.

Descripción: Dentro de los ensayos no destructivos (NDT), las ondas guiadas son reconocidas como una práctica muy eficiente para el estudio del monitoreo de salud estructural ya que permite abarcar longitudes mucho mayores que las prácticas de ultrasonido convencional y acceder a zonas de difícil acceso. Uno de los principales problemas que se encuentran en el uso de esta tecnología, es encontrar una frecuencia de sintonización apropiada que sea capaz de propagarse en el material a larga distancia y poder interferir con discontinuidades en la sección transversal de la estructura. Las curvas de dispersión describen en qué modos se va a descomponer una excitación en la estructura, conocer estas curvas permite conocer el estado o condición de la estructura. El método convencional para la generación de curvas de dispersión es a partir de una simulación en un software de elementos finitos, lo que genera datos de las frecuencias naturales y modos de vibración de la estructura con las condiciones establecidas en el modelo, luego, se hace un postprocesamiento de toda la información modal en otro software para poder identificar, clasificar y relacionar todos los modos, lo que hace que sea una metodología compleja y tediosa.

En este trabajo se presenta el modelo para la generación de curvas de dispersión en placas planas, cilindros y perfiles en I de materiales isotrópicos, y el desarrollo de una aplicación en entorno Matlab para la generación de curvas de dispersión en placas planas, así como la generación y validación de curvas de dispersión en placas planas con las curvas generadas por el software GUIGW.

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Mecánica. Director: Jabid Eduardo Quiroga Méndez. PhD Ingeniería Civil.

Abstract

Title: Development of an application in Matlab for the generation of dispersión curves of guided waves in flat plates, tubes and I profiles of isotropic materials.[*]

Author: Enrique Ortega Pinzón[**]

Key Words: Dispersion curves, Finite elements, PDE Matlab.

Description: Within non-destructive testing (NDT), guided waves are recognized as a very efficient practice for the study of structural health monitoring since it allows covering much greater lengths than conventional ultrasound practices and accessing areas of difficult access. One of the main problems found in the use of this technology is to find an appropriate tuning frequency, able to propagating in the material over a long distance and being able to interfere with discontinuities in the cross section of the structure. Dispersion curves describe in which ways an excitation is going to be decomposed in the structure. Knowledge of these curves allows us to know the state or condition of the structure. The conventional method for the generation of dispersion curves is from a simulation in a finite element software, which generates data on the natural frequencies and modes of vibration of the structure with the conditions established in the model, then, it is done a post-processing of all modal information in other software to be able to identify, classify and relate all the modes, which makes it a tedious and complex methodology.

In this work the model for the generation of dispersion curves in flat plates, cylinders and I-profiles of isotropic materials is presented, and the development of an application in Matlab environment for the generation of dispersion curves in flat plates, as well as the generation and validation of flat plate dispersion curves with the curves generated by the GUIGW software.

---

[*] Degree Work

[**] Faculty of Physical-Mechanical Engineering. Department of Mechanical Engineering. Director: Jabid Eduardo Quiroga Méndez. PhD Civil Engineering

Introducción

El aumento continuo en los costes de fabricación, condiciones de operación más extremas para los componentes estructurales y la tendencia de un diseño hacia un ajuste más estrecho de los márgenes de seguridad ha impulsado una constante evolución y avance en los métodos de ensayos no destructivos (NDT). A partir de los resultados de los NDT aplicados a un elemento, se puede evaluar la seguridad ante un posible fallo, realizar predicciones de la vida útil restante y gestionar el plan de mantenimiento adecuado para la situación.

Las Técnicas de Ondas Guiadas se han convertido en un tema de gran relevancia dentro de los NDT y el Monitoreo de Salud Estructural (SHM) durante la última década. La inspección por ondas guiadas permite la detección de la degradación de la estructura, reduce los costos de acceso y evita la remoción o reposición generalizada del material aislante o revestimiento, inspeccionando grandes distancias de estructura desde su punto de aplicación, lo que reduce los costos y tiempos de inspección.

El método convencional para la generación de curvas de dispersión es a partir de un análisis modal de la pieza estudiada en un software especializado en elementos finitos y exportar la información modal un software para poder hacer la identificación, clasificación y relación de cada uno de los modos encontrados en el análisis de elementos finitos. A esto se le debe sumar la extensa investigación que debe ser realizada. Todo esto, hace que el método convencional del estudio de la propagación de ondas guiadas sea lento, complejo y tedioso. Por lo que es necesario buscar una alternativa para la automatización del cálculo y generación de las curvas de dispersión para buscar mejorar la eficiencia del método.

En este trabajo se presenta el modelo para la generación de curvas de dispersión en placas planas, tubos y perfiles en I de materiales isotrópicos, el desarrollo de una aplicación en entorno Matlab para la generación de curvas de dispersión en placas planas mediante el uso de la PDE toolbox para el desarrollo del modelo de elementos finitos haciendo el postprocesamiento de la información dinámica a partir de algoritmos y posteriormente, se hace la validación de varias curvas generadas con la aplicación con el software GUIGW. También se presentan las bases para el desarrollo de una aplicación en Matlab para la generación de curvas de dispersión en diferentes geometrías para futuros trabajos.

1. Objetivos

1.1.Objetivo General

Desarrollar una aplicación en ambiente Matlab que genere las curvas de dispersión (velocidad de fase, velocidad de grupo y frecuencia) de las ondas guiadas en placas planas, tubos y perfiles en I de materiales isotrópicos a partir de un sólido generado en un software de diseño CAD teniendo en cuenta las propiedades del material.

1.2.Objetivos Específicos

- Calcular las frecuencias naturales y modos de vibración de guías de onda en placas planas, tubos y perfiles en I a través del uso de la herramienta de elementos finitos de Matlab PDE toolbox.
- Post-procesar la información dinámica obtenida en el objetivo anterior, para, a partir de esto y de la geometría del sólido, generar las curvas de dispersión.
- Validar las curvas obtenidas con las disponibles en la literatura.
- Diseñar una interfaz que permita la generación de las curvas de dispersión a partir del sólido generado en un software de diseño CAD.

2. Ondas guiadas en NDT y SHM

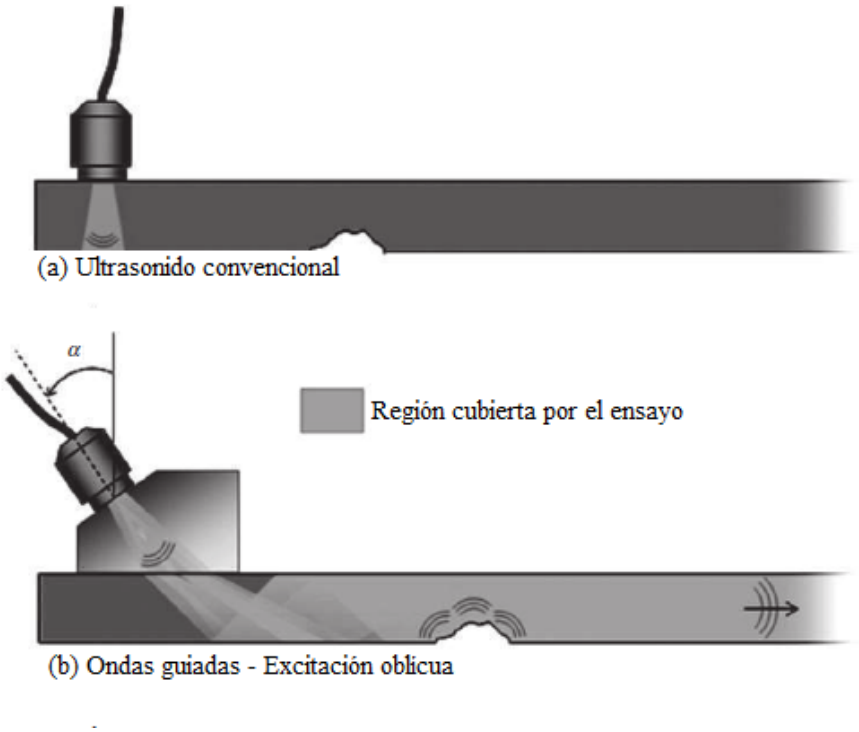
Un sistema con SHM está conformado, esencialmente, por una red de sensores equipados en muchas ubicaciones de la estructura bajo inspección para la adquisición de datos, que, en conjunto con un sistema de procesamiento, evalúan la salud estructural. La capacidad de identificar posibles fallas o daños estructurales aumenta la confiabilidad de la estructura o el equipo, garantiza la seguridad del personal y reduce el tiempo de inactividad, lo que reduce los costos de mantenimiento y operación.

Las ondas guiadas ultrasónicas se han convertido en un tema de gran relevancia dentro de los NDT y SHM. Gracias a esta técnica, se han hecho posible nuevas formas más rápidas, más sensibles y económicas de observar los materiales y las estructuras en comparación con las técnicas de inspección de ultrasonido convencionales. Por ejemplo, el proceso de inspección de una tubería aislada requería quitar todo el aislamiento y usar una sola sonda para verificar con un haz normal a lo largo de la tubería con miles formas de onda. Ahora, se puede usar una sonda de onda guiada en un solo lugar e inspeccionar toda la tubería examinando solo algunas formas de onda dejando el aislamiento intacto. (Rose, 2014)

Las ondas guiadas son ondas ultrasónicas que se propagan a lo largo de una estructura, guiadas por sus límites. En el caso de una estructura alargada como una viga o placa, la onda guiada puede viajar durante mucho tiempo con atenuación relativamente baja, lo que lo hace útil para propósitos de NDT (Dunn, 2018). Para inspeccionar el estado de una estructura larga, en el método de ultrasonido convencional, la ubicación del transductor debe cambiarse junto con la estructura, la onda se propaga en un área limitada y con una atenuación muy rápida, por lo que el rango de sensibilidad del método es limitado (Zhang et al., 2018).

Figura 1

Comparación entre el alcance de un ensayo por ultrasonido convencional y ondas guiadas. (Rose,2014)



En la Figura 1 se puede observar cómo la técnica de ondas guiadas consigue un mayor alcance debido a su incidencia oblicua. El potencial de las ondas guiadas para estructuras con una dimensión mucho mayor que las otras, como son, ductos, vías de tren, perfiles estructurales, entre otros. En el caso de los rieles y conductos, por ejemplo, estas estructuras sufren cargas y descargas cíclicas, en el caso del riel por el paso del tren y en el caso del conducto por la variación de la presión (Grot, 2016). Todos estos esfuerzos generan fatiga y una posterior fractura. En el caso de las tuberías, otro aspecto crítico es la corrosión, la cual puede ser severa dependiendo del medio en el que se sumerja la tubería y el tipo de fluido transportado. Todo esto genera un gran interés

en la evaluación y monitoreo de estas estructuras, y estos son solo algunos ejemplos de una gran variedad de aplicaciones encontradas para NDT y SHM basados en ondas guiadas.

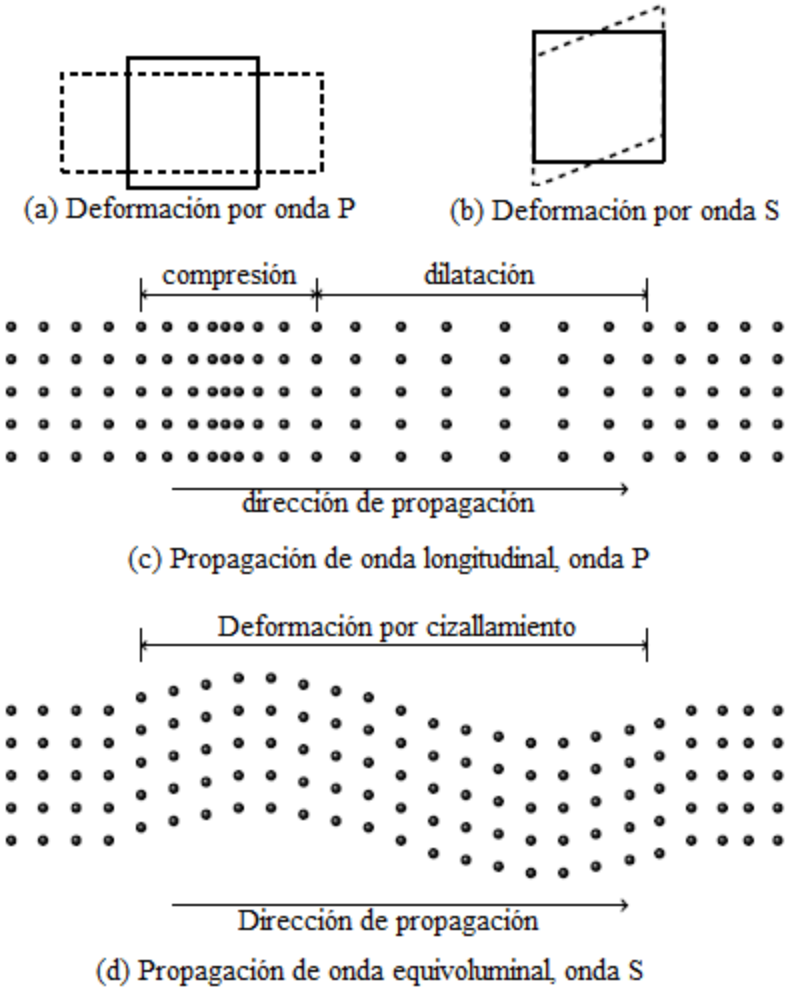
3. Propagación de ondas en un medio sólido

La acústica es el estudio de las formaciones o vibraciones que varían en el tiempo en el entorno del material. Los átomos que forman el medio material oscilan alrededor de su posición de equilibrio cuando el medio material es sometido a un movimiento vibratorio. En el estudio de la acústica se considera que el material está formado por pequeñas partículas, pero cada partícula contiene varios átomos que se mueven al unísono, Por tanto, la acústica puede ser considerada un fenómeno macroscópico en el que la materia se modela como un medio continuo (Auld, 1973).

Este medio continuo no presenta discontinuidades entre sus partículas, de forma que se puede realizar una descripción matemática de sus propiedades físicas empleando funciones continuas (Rose, 2014). En presencia de una perturbación externa, se generan ondas que se propagan a través de elementos discretos que simulan pequeños cuerpos másicos con propiedades elásticas. Hay dos tipos de movimientos básicos que puede experimentar un cubo infinitesimal dentro del medio elástico al producirse el paso de la onda sobre sí, un movimiento de dilatación/compresión y un movimiento de distorsión. El primero altera el volumen sin cambiar los ángulos rectos en sus bordes, se conoce como como onda dilatacional o P (primaria). En el movimiento de distorsión, el volumen permanece constante mientras se distorsiona, se le conoce como onda equivoluminal o S (secundaria) (Sofer 2018).

Figura 2

Posibles movimientos de las partículas que componen un medio infinito debido al paso de una onda de tensión. (a) y (b) son deformaciones por el cubo infinitesimal. (c) y (d) son el movimiento de las partículas sólidas durante el paso de la onda



3.1 Reflexión de ondas mecánicas

Las ondas pueden encontrar un cambio en el entorno en el que su propagación se reflejará en la interfaz o será transmitida a otro medio, también puede cambiar su modo de propagación:

puede reflejarse o refractarse al interactuar con una interfaz. En el caso de la propagación en un medio sólido, el comportamiento se rige por la Ley de Snell.

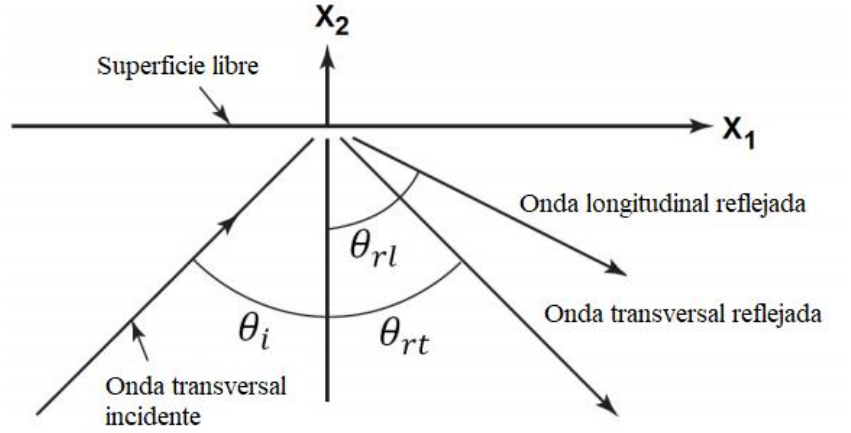
Una onda transversal que incide sobre una superficie reflectora se separará en dos ondas reflejadas, una transversal y una longitudinal, reflejadas respectivamente en un ángulo de θ_{rt} y un ángulo θ_{rl}, en el que el segundo depende de las velocidades de propagación c_1 y c_2. Los dos tipos de onda satisfacen la ecuación (1):

$$\frac{\sin(\theta_{rl})}{c_1} = \frac{\sin(\theta_i)}{c_2} \tag{1}$$

Donde $c_1$es la velocidad de propagación de la onda longitudinal en el medio y c_2 es la velocidad de propagación de la onda transversal en el mismo medio. La Figura 3 muestra el comportamiento de reflexión de ondas en medios sólidos (Idzi 2017).

Figura 3

Reflexión de ondas en un medio sólido según la ley de Snell (Eagle, 2005).



En la Figura 3 se puede observar que la onda longitudinal reflejada será normal a la onda incidente transversal, es decir:

$$\theta_{rl} + \theta_i = 90 \quad (2)$$

En el caso de que la onda incidente sea longitudinal, se pueden encontrar expresiones similares para determinar el ángulo de reflexión de las ondas reflejantes.

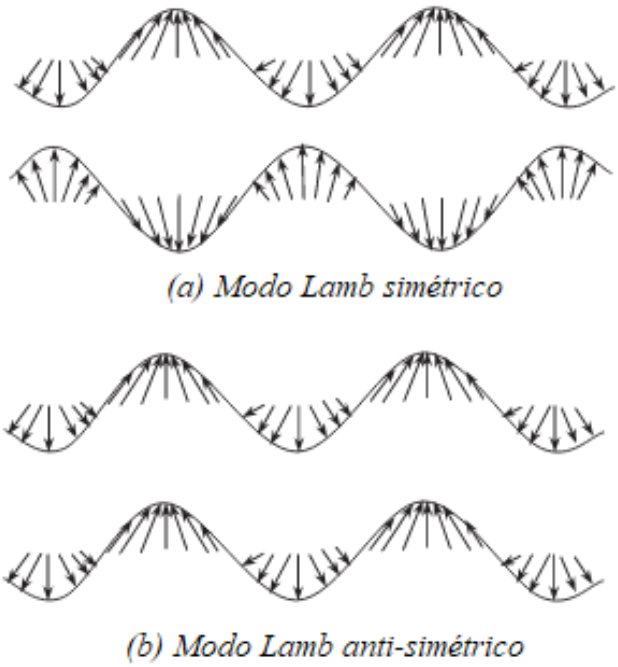
3.2 Ondas Lamb

Las ondas Lamb aparecen en sólidos donde las ondas S y P son guiadas entre dos superficies paralelas. La interferencia entre las múltiples reflexiones produce ondas que se propagan paralelamente a las superficies. Los diferentes patrones de interferencia dan lugar a un número infinito de modos Lamb, cada uno con un perfil de desplazamientos según el espesor. Dependiendo de la frecuencia de excitación, las ondas Lamb son producidas en por lo menos dos modos básicos, modos simétricos y modos antisimétricos. Las deformaciones en las placas planas se dan de forma que los lados superior e inferior se alejan del centro de la placa (o hacia ellos) simultáneamente. En el caso de los modos antisimétricos la parte superior se aleja o se acerca del centro de la placa mientras que la parte inferior se mueve hacia la misma dirección que la parte superior. En la figura 4 se muestra el comportamiento de los modos simétricos y antisimétricos.

La velocidad de propagación depende del producto de la frecuencia por el espesor de la placa, por lo cual, estas ondas son dispersivas (Zhongqing y Lin, 2006). Esto significa que existe una relación entre la longitud de onda o el número de onda con la frecuencia. A partir de esta relación, la velocidad de fase y la velocidad de grupo de la onda están determinadas por una relación de dispersión dependiente de la geometría y del material.

Figura 4

Deformación de los modos simétricos y anti-simétricos (Zhongqing y Lin, 2006).



La excitación ultrasónica ocurre en algún punto de la placa, a medida que la energía ultrasónica de la región de excitación se encuentra con las superficies límite superior e inferior de la placa, se producen conversiones de modo (onda L a onda T y viceversa). Después de algún viaje en la placa, las superposiciones forman "paquetes de ondas", o lo que se le llama comúnmente modos de ondas guiadas en la placa. Basándose en el ángulo de entrada y la frecuencia utilizadas, es posible predecir cuántos modos diferentes se pueden producir en la placa (Rose y Nagy, 2002).

3.2.1 Solución de modos de Lamb en placas

La solución para los modos de Lamb se obtiene a través las ecuaciones de Rayleigh-Lamb. Las curvas de dispersión muestran todas las posibilidades de modos propagadores (maneras de propagar) que puede tener la placa, funcionando como un mapa de propagación.

En una estructura específica, las curvas de dispersión de una estructura son límites que no serán sobrepasados, es decir, que nunca habrá una onda que se propague con un par (w, k) que esté fuera de las líneas de las curvas de dispersión (Rose, 2003). Las ecuaciones de Rayleigh-Lamb se muestran a continuación:

$$p^2 = \left(\frac{\omega}{c_L}\right)^2 - k^2 \quad ; \quad q^2 = \left(\frac{\omega}{c_T}\right)^2 - k^2 \tag{3}$$

$$\frac{\tan(qh)}{\tan(ph)} = -\frac{4k^2pq}{(q^2-k^2)^2} \tag{4}$$

$$\frac{\tan(qh)}{\tan(ph)} = -\frac{(q^2-k^2)^2}{4k^2pq} \tag{5}$$

Donde la expresión (4) es para los modos simétricos y la expresión (5) para los modos antisimétricos. El número de onda k es numéricamente igual a ω/c_p, donde c_p es la velocidad de fase del modo de onda Lamb y ω es la frecuencia circular. La velocidad de fase está relacionada con la longitud de onda por la relación $c_p = (\omega/2\pi)\lambda$ (Rose, 2014).

3.3 Aspectos generales de las Ondas

En un medio definido hay dos tipos fundamentales de ondas: una longitudinal y una transversal. Si el medio es finito, los fenómenos de reflexión surgen en los bordes. En una placa o tubo finito, las ondas longitudinales y transversales se reflejan y recogen sucesivamente generando una propagación de una forma particular. Las ondas son generadas por dispositivos adecuados y pueden describirse mediante una suma de armonías. En una placa, pueden existir varias ondas armónicas a una sola frecuencia dada, llamados modos. En la práctica, solo se utilizan modos particulares que sean fáciles de generar y medir.

Una onda progresiva unidimensional puede ser descrita analíticamente por:

$$\Psi(x,t) = A\cos(\omega t - kx) \tag{6}$$

Donde A es la amplitud, ω es la frecuencia angular, t es el tiempo y k el número de onda. Las cuales siguen las siguientes relaciones:

$$\omega = 2\pi f = \frac{2\pi}{T}; \qquad k = \frac{2\pi}{\lambda} \tag{7}$$

4. Extracción numérica de curvas de dispersión

En la literatura se puede observar que una de las técnicas de extracción de curvas de dispersión más utilizadas es mediante la hibridación con un código de elementos finitos. Una de las metodologías utilizadas para determinar las curvas de dispersión es a partir de formulaciones SPBW (superposition of bulk waves), estas proporcionan la solución exacta de propagación de ondas guiadas en placas, barras o cilindros simétricos. Dentro de esta metodología se han utilizado técnicas matriciales como el Método Transfer Matrix (TMM) y el Método Global Matrix (GMM) (Lowe, 1995). En los métodos SPBW, la ecuación de onda toma la forma de un problema de valor propio trascendental no lineal.

Otro método es el SAFE (Semi-Analytical Finite Element), el cual hace uso de una malla discrita unidimensional o bidimensional para describir el campo de desplazamiento de onda sobre la sección transversal de la guía de onda y transformadas de Fourier para condensar el problema de onda en la dirección de propagación y en el tiempo.

También se han empleado enfoques de Elementos Finitos (EF) para extraer el espectro dispersivo de guías de onda. Si bien, obtener las curvas de dispersión de un paquete de EF general no es sencillo, ya que requiere una gran cantidad de análisis modal de guías de onda de sección

transversal uniforme, resulta en un proceso que consume mucho tiempo, en comparación con las formulaciones SAFE, tiene la ventaja de que se pueden utilizar paquetes de EF estándar.

Sorohan et al (2011) propone un enfoque numérico en el que, a partir de un análisis dinámico (frecuencias naturales y modos de vibración), se establece una relación número de onda-frecuencia, siendo esta, la base de las curvas de dispersión. Una de las mayores ventajas de este modelo EF es que es capaz de determinar las curvas de dispersión de estructuras complejas.

En el presente trabajo se obtendrán las curvas de dispersión mediante el modelo presentado por Sorohan et al (2011), el cual parte de un modelo de EF y, a partir de la información dinámica que da como solución, se hace un análisis de cada uno de los modos para determinar el tipo de modo para así poder hacer una clasificación y relación de los modos y generar las curvas de dispersión.

4.1 Modelo de Elementos Finitos

El método de elementos finitos consiste en dividir, se debe dividir la estructura en un número de elementos finitos. Dentro de cada elemento fino se asume una formulación polinomial y se ensambla en un sistema más grande de ecuaciones que modela todo el problema.

Para el caso de las guías de onda, se debe discretizar con una malla para calcular las frecuencias naturales y los modos de vibración, no se considerarán cargas aplicadas en el sistema, es decir, se considera una situación de “espacio libre”. Para determinar las frecuencias naturales y los modos de vibración, el análisis se basa en el movimiento no forzado gobernado por la ecuación de un conjunto de dos ecuaciones diferenciales de segundo orden acopladas, que se escribe en forma matricial como:

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]u = \{F\} \quad (8)$$

Donde [$\boldsymbol{M}$] es la matriz de masa, [**C**], es la matriz de amortiguación, [$\boldsymbol{K}$] es la matriz de rigidez, los vectores de desplazamiento, velocidad y aceleración $\boldsymbol{u}(t)$, $\dot{\boldsymbol{u}}(t)$ y $\ddot{\boldsymbol{u}}(t)$. El vector $\boldsymbol{f}(t)$ está definido por las fuerzas asociadas con cada grado de libertad. Si la amortiguación y las fuerzas aplicadas se ignoran en el análisis estructural, la ecuación de movimiento (6) se transforma en una ecuación que describe las vibraciones libres.

$$[M]\{\ddot{u}\} + [K]u = 0 \quad (9)$$

La solución de esta ecuación tiene la forma:

$$\{u\} = \{\phi\}e^{i\omega t} \quad (10)$$

Donde $\{\phi\}$ es el modo de vibración, ω es la eigen-frecuencia angular. Incluyendo la solución y la segunda derivada en la ecuación (9) da como resultado:

$$(-\omega^2[M] + [M]\,)\{\phi\} = \{0\} \quad (11)$$

Este problema de eigenvalores generalizados tiene n pares de eigenvalores ω_j^2 y eigenvectores asociados $\{\phi\}$.

5. Metodología

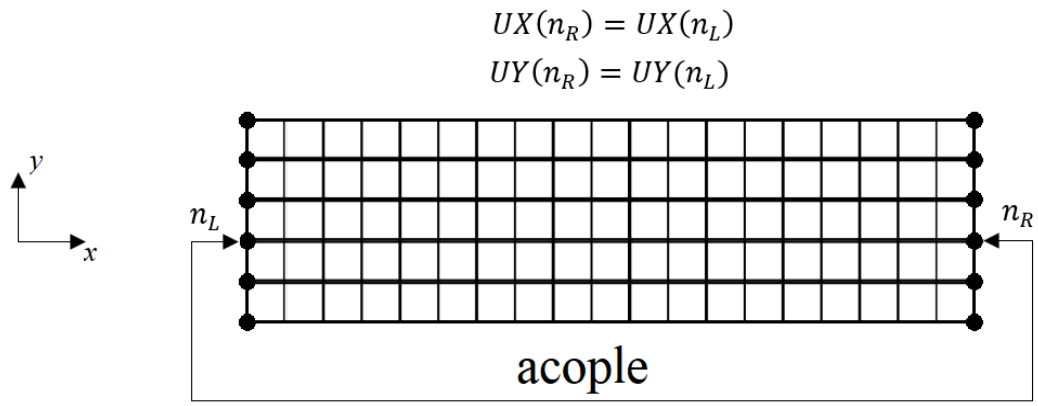
Para las placas, el modelo de FE apropiado consiste en un rectángulo teniendo como base la longitud de la placa (L) y la altura (h). Sorohan et al. (2011) resalta que para una buena aproximación la relación de la longitud con la altura debe ser $L = (10 - 20)h$. Este modelo debe ser uniformemente mallado con elementos definidos con 2 DOFs en cada nodo: desplazamientos en direcciones *X* e *Y*.

Según el procedimiento propuesto por Sorohan et al. (2011), para obtener los modos Lamb y determinar el comportamiento de la guía de onda, se requiere un acople mecánico entre el

extremo izquierdo y el extremo derecho de la sección transversal de la placa. Es decir, para cada par de nodos simétricos (n_L y n_R) en los extremos derecho e izquierdo del rectángulo, los DOFs vertical y horizontal son acoplados, lo que hace que los desplazamientos en *X* e *Y* sean iguales: $UX(n_L) = UX(n_R)$ y $UY(n_L) = UY(n_R)$. Esta condición de simetría es también llamada "Condiciones de Contorno Periódicas" (PBC).

Figura 5

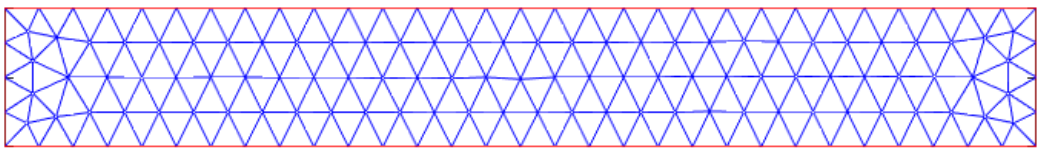
Modelo de FE usado para el análisis modos Lamb en placas.



Para la solución del modelo de FE en Matlab se utiliza la PDE toolbox, la cual proporciona funciones para resolver la mecánica estructural y las ecuaciones diferenciales generales mediante el análisis de elementos finitos (Mathworks 2020). Esta toolbox permite importar geometrías 2D y 3D en formato STL o datos de malla. Puede generar mallas con elementos triangulares y tetraédricos.

Figura 6

Ejemplo de mallado generado con PDE toolbox en Matlab



A pesar de que la PDE toolbox facilita el análisis de modelos de FE, las condiciones de contorno periódicas (PBC) no son compatibles con su flujo de trabajo, lo que significa que no existe una instrucción o herramienta directa para aplicarlas. La única forma de aplicar las PBC utilizando la PDE toolbox es extrayendo las matrices del modelo FE y modificándolas para imponer las condiciones "manualmente". Esto último es posible con la instrucción assembleFEMatrices incluida en la PDE toolbox. Al establecer el modelo FE como estructural y un análisis modal como parámetros de entrada en el modelo de PDE, carga las matrices necesarias para el análisis, luego, con la instrucción "assembleFEMatrices" es posible modificar la matriz de rigidez y la matriz de masa para imponer las condiciones de contorno periódicas. Para la solución del modelo se usa la instrucción "eigs" con parámetros de entrada las matrices modificadas de rigidez y masa. El programa para el análisis modal se puede observar en el Apéndice B.

Como solución del modelo FE, se obtienen las eigen-frecuencias y los eigen-modos. Todos las frecuencias y modos aparecen en pares, esto indica que las dos ondas se están propagando simultáneamente en direcciones opuestas. Por esto, es necesario hacer un filtro para eliminar las frecuencias y modos que se repiten.

5.1 Post-procesamiento

A partir de la información de la dinámica estructural que da como solución el análisis modal, para poder generar las curvas de dispersión es necesario calcular la longitud de onda y clasificar en simétrico o antisimétrico cada modo.

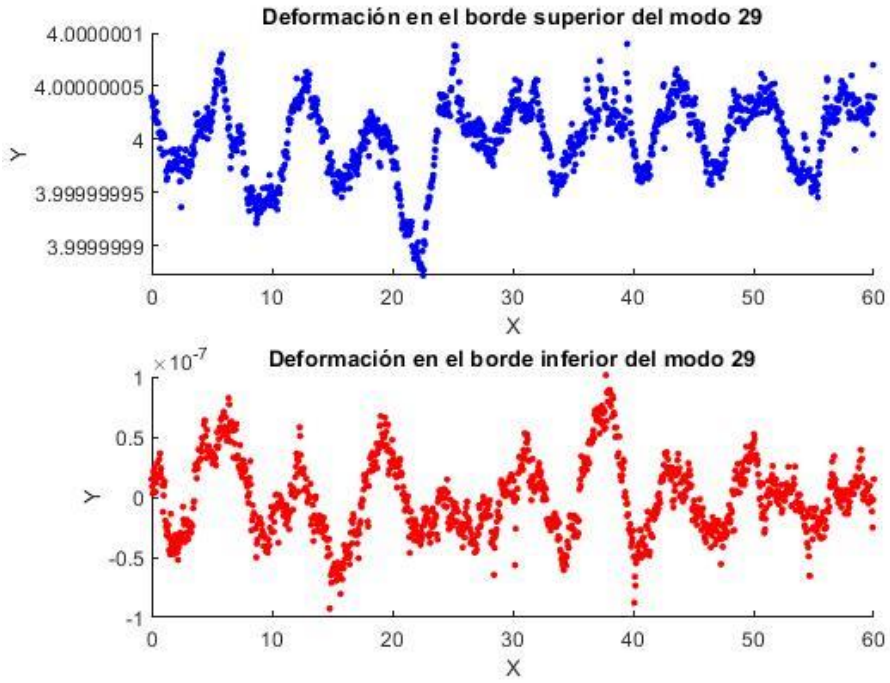
5.1.1 Clasificación de modos armónicos

Como se mencionó anteriormente, los modos y frecuencias de la solución del análisis modal aparecen en pares, sin embargo, en la solución de Matlab aparecen modos que no tienen un par. Graficando estos modos se observa que tienen un comportamiento extraño, no armónico como

se observa en la Figura X. Si bien esto puede ser porque Matlab no está especializado con modelos de FE y estos modos aparecen como producto de una aproximación, también pueden ser modos que producen resonancia en la placa en una dirección diferente a la guía de onda.

Figura 7

Deformación en los bordes superior e inferior de un modo no armónico



Entonces, la función para filtrar los modos no solamente debe eliminar los modos que están repetidos, sino también los modos que no tienen un par. Para esto, se crea un ciclo iterativo que compare cada frecuencia con la siguiente, de modo que pueda identificar cuando un modo está sin un par y lo elimine, y cuando 2 modos son el mismo, y elimine uno. Este algoritmo se puede observar como una función en el Apéndice D.

5.1.2 Clasificación de modos por el número de modo (p)

Esta clasificación consiste en una relación de la longitud de onda del modo con la longitud de la placa, es decir, los valores de p que satisfacen la relación:

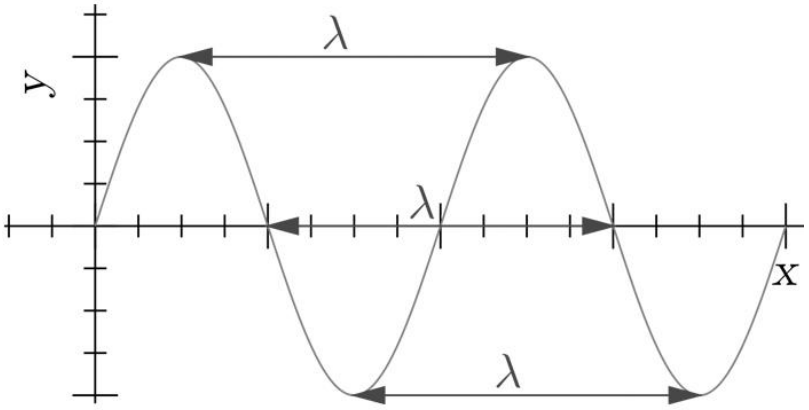
$$\lambda = \frac{L}{p} \tag{12}$$

Donde L es la longitud de la placa y λ es la longitud de onda.

Sabiendo que la longitud de onda es la distancia que hay entre 2 crestas o 2 valles consecutivos como se muestra en la Figura 8, es posible determinar la longitud de onda a partir de los desplazamientos producidos por cada modo.

Figura 8

Longitud de onda



A partir de los desplazamientos de la solución del análisis modal, el algoritmo desarrollado calcula la pendiente de la recta que se produce con las coordenadas de un nodo con el siguiente. El algoritmo calcula la pendiente entre los 2 primeros nodos y la compara con la pendiente de los siguientes 2 nodos en un bucle hasta comparar todos los nodos. Cuando hay un cambio en el signo de la pendiente el algoritmo lo registra como un valle o una cresta: una cresta cuando cambia de

una pendiente positiva a negativa y un valle cuando cambia de una pendiente negativa a una positiva. Al final, el algoritmo contará cuantos valles y crestas produce la onda en la longitud de la placa. El número total de valles o crestas será igual a *p.* Este algoritmo se puede observar como una función en el Apéndice F.

5.1.3 Clasificación de modo en A o S

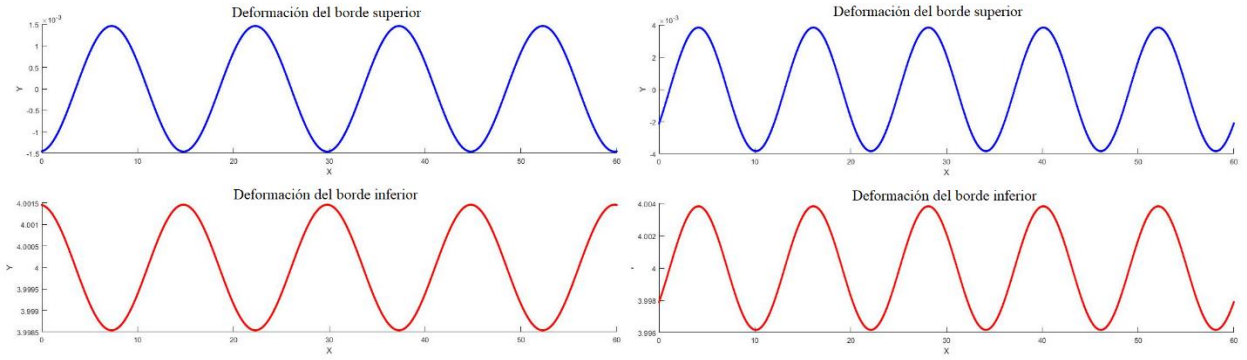
En su procedimiento, Sorohan et al. (2011) utiliza el Criterio de Aseguramiento Modal (MAC) para determinar los pares de modos correlacionados. Sin embargo, al intentar aplicar esta metodología en Matlab los resultados no fueron los esperados. Esto puede ser porque no hay la sensibilidad necesaria en los valores de las frecuencias y los modos, ya que como se dijo anteriormente, Matlab no está especializado en modelos FE. Por esto, para la identificación de los modos, se desarrolló un algoritmo que clasifica cada modo en simétrico (S) o antisimétrico (A).

Para el desarrollo del algoritmo se analizó la forma de los modos simétricos y antisimétricos. Las deformaciones en las placas planas se dan de forma que los lados superior e inferior se alejan del centro de la placa (o hacia ellos) simultáneamente. En el caso de los modos antisimétricos la parte superior se aleja o se acerca del centro de la placa mientras que la parte inferior se mueve hacia la misma dirección que la parte superior, como se observa en la Figura 4.

En la Figura 10 se muestra un modo simétrico y un modo antisimétrico obtenidos con Matlab.

Figura 9

Desplazamientos de modos. a) Desplazamientos en el borde superior e inferior de un modo simétrico. b) Desplazamientos en el borde superior e inferior de un modo antisimétrico



a) b)

El algoritmo para clasificar el modo en simétrico o antisimétrico consiste en calcular la pendiente entre los 2 primeros nodos y la compara con la pendiente de los siguientes 2 nodos en un bucle hasta llegar a la coordenada de la primera cresta o valle. Esto lo hace tanto para los nodos en el borde inferior como para el superior. Luego compara el comportamiento de las pendientes en los 2 bordes. Si las pendientes tienen el mismo signo, entonces el modo es antisimétrico. Si las pendientes son opuestas, entonces el modo es simétrico. En el Apéndice E se muestra el algoritmo para clasificar los modos en A o en S.

5.1.4 Cálculo de la velocidad de fase y la velocidad de grupo

Para obtener los datos finales para la generación de curvas de dispersión, es necesario realizar algunos cálculos. El número de onda *k* está relacionado con la longitud de onda λ como se observa en la Ecuación 13

$$k = \frac{2\pi}{\lambda} \tag{13}$$

La velocidad de fase puede ser calculada fácilmente a partir de los datos de la velocidad angular:

$$\omega = 2\pi f \tag{14}$$

$$C_{ph} = \lambda * v = \frac{\omega}{k} \tag{15}$$

La velocidad de grupo está definida como la derivada de la frecuencia respecto al número de onda:

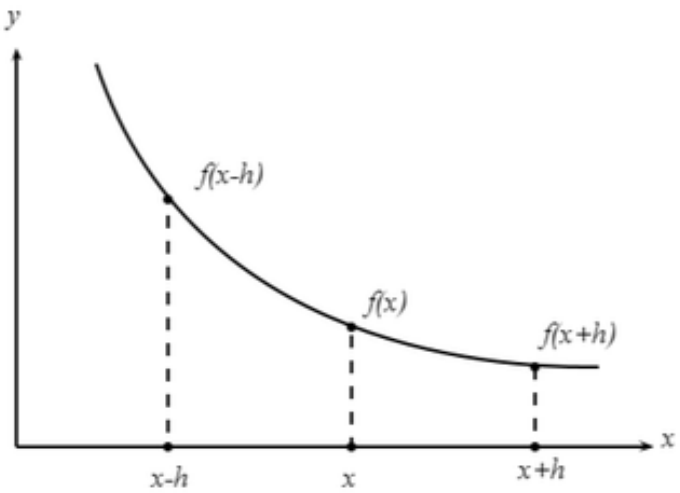
$$C_G = \frac{d\omega}{dk} \tag{16}$$

Sorohan et al. (2011) propone realizar este cálculo utilizando el método de la diferencia central para estimar cada coordenada (C_G, f). Para el método de diferencia central se hará un análisis modal con una nueva longitud *L* con una perturbación ΔL entre (0.5-2.5)% de *L*. De esta manera, el cálculo de la velocidad de grupo queda de la forma:

$$\frac{df(x)}{dx} = \frac{f(x+h) - f(x-h)}{2h} \tag{17}$$

Figura 10

Representación método de diferencias finitas. Recuperado de: http://mmc.geofisica.unam.mx/



6. Modelo de FE para la generación de curvas de dispersión en tubos y perfiles en I

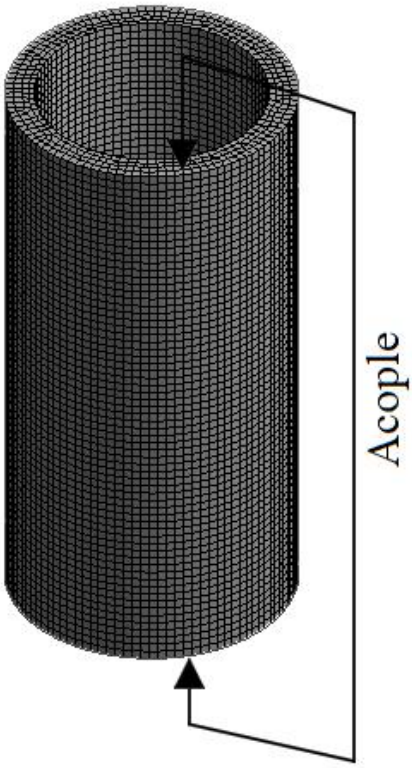
El modelo FE que se presentará a continuación consiste en calcular las frecuencias naturales y modos de vibración de la guía de onda colocando condiciones de contorno periódicas

simétricas y antisimétricas de moto a permitir que aparezcan los modos correspondientes a las ondas modales buscadas.

Las condiciones de contorno periódicas son aplicadas en las 2 caras extremas de la sección transversal. De modo que cada par de nodos simétricos en los extremos de la geometría, los DOFs en X, Y y Z son acoplados, lo que hace que los desplazamientos sean iguales.

Figura 11

Acople en tubo.

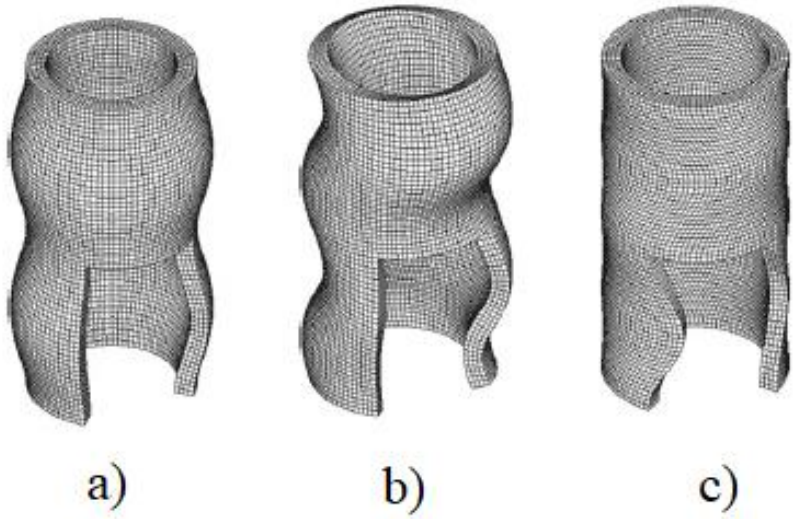


En comparación con las placas, existen más tipos de modos Lamb en los tubos y perfiles en I. Los modos más comúnmente generados son los modos Longitudinales, Flexurales y Torsionales. En la Figura 12 se observa el comportamiento de los modos longitudinales, Flexurales

y Torsionales en un tubo. En un perfil en I se presentan los mismos modos que en un tubo, y se comportan de la misma manera

Figura 12

Modos Lamb en tubo. a) Modo Longitudinal. b) Modo Flexural. c) Modo Torsional



Como se mencionó anteriormente, la PDE toolbox de Matlab no tiene una instrucción o una herramienta para aplicar las condiciones de contorno periódicas porque el flujo de trabajo no es compatible con Matlab. Al igual que con el modelo de FE de las placas, para aplicar estas condiciones es necesario extraer las matrices del modelo con la instrucción “assembleFEMatrices” y modificarlas, de forma que se puedan imponer las condiciones de simetría y antisimetría. A diferencia que el modelo FE de las placas, en los tubos y perfiles en I se considerarán los 3 DOFs para las condiciones de contorno, X, Y e Z. Por lo anterior, para la identificación del modo es necesario analizar tanto en el plano XY como en el plano YZ. En el plano XY se presentan los modos Longitudinales y Flexurales, mientras que en el plano YZ se presentan los modos Torsionales.

Si bien se logró codificar el análisis modal con las condiciones de contorno periódicas en 3 DOFs, se presentan inconvenientes al momento de analizar gráficamente cada modo, lo que impide su identificación. Trabajando en conjunto con el equipo de soporte oficial de Mathworks se concluyó que se debe a la incompatibilidad de la PDE toolbox con las PBC, por lo que no se pudo desarrollar un programa para la generación de curvas de dispersión en tubos y en perfiles en I.

En el Apéndice H se adjunta el código en Matlab para el análisis modal con condiciones de contorno periódicas con 3 DOFs, el cual queda como base para futuros trabajos.

7. Resultados

De manera que se pueda comprobar la eficiencia del programa, se grafican las curvas generadas en 2 ejemplos con el programa en Matlab y los resultados dados por el software GUIGW para comparar y validar el método propuesto y el programa desarrollado,

Como primer ejemplo se graficarán las curvas de una placa de aluminio de 4 mm de espesor y 60 mm de longitud, con módulo de Young $E = 70\ GPa$, coeficiente de Poison $v = 0.33$ y densidad $\rho = 2700\ kg/m^3$.

Figura 13

Longitud de Onda Vs Frecuencia. Ejemplo 1

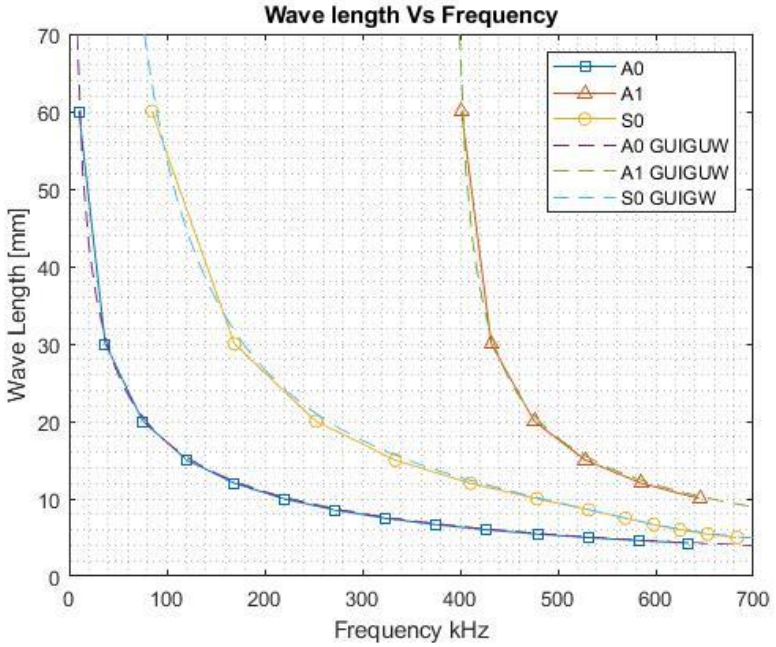


Figura 14

Número de onda Vs Frecuencia. Ejemplo 1

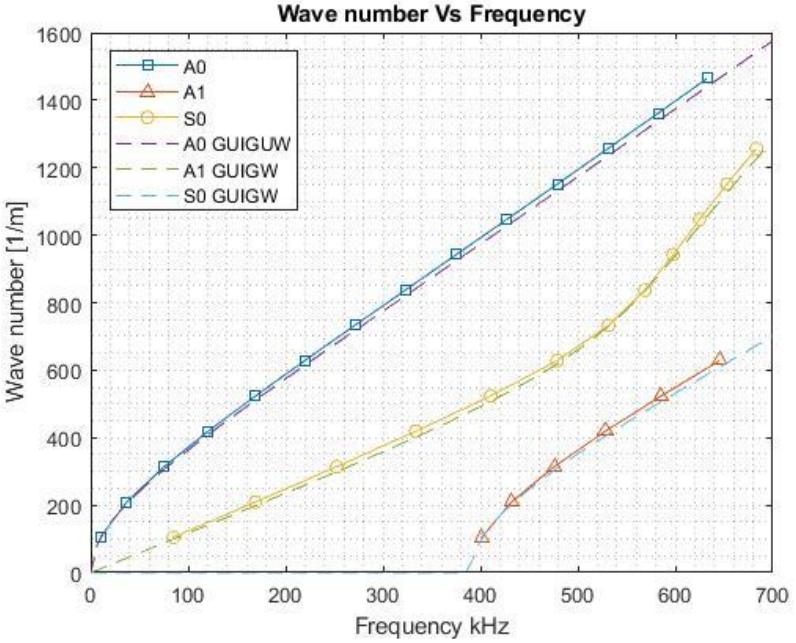


Figura 15

Velocidad de fase Vs Frecuencia. Ejemplo 1

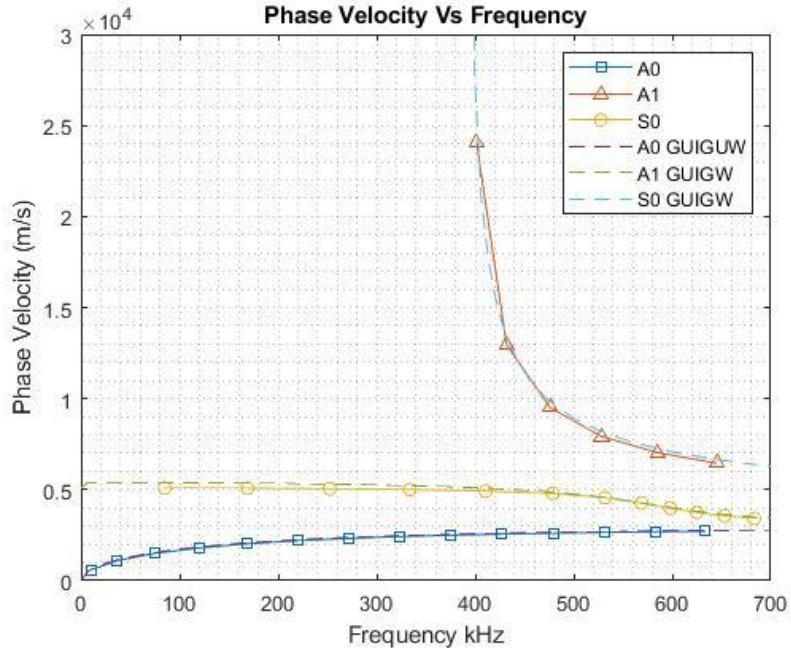
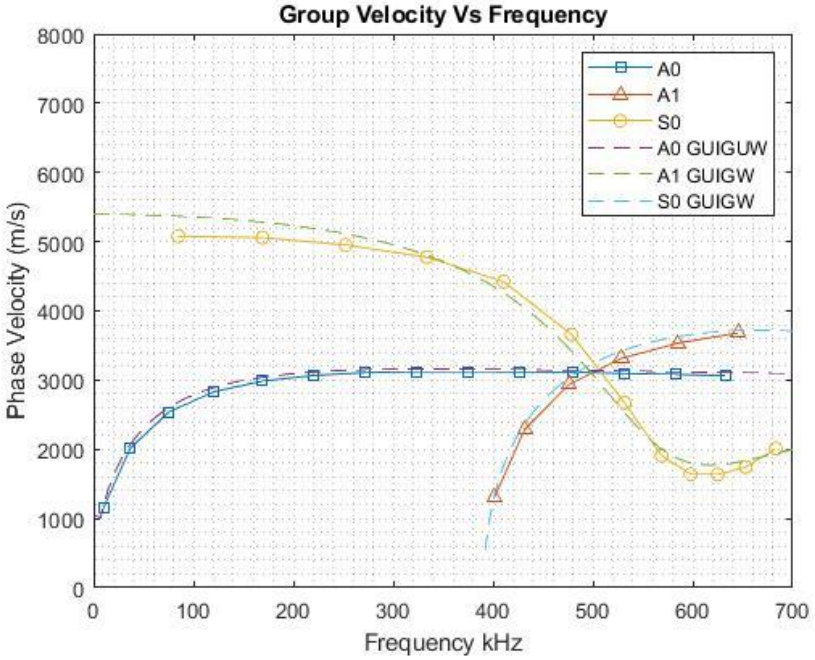


Figura 16

Velocidad de Grupo Vs Frecuencia. Ejemplo 1



Como segundo ejemplo se generarán las curvas de dispersión de una placa de Aluminio de 5 mm de espesor y 90 mm de longitud. ($E = 70e9\ Pa, v = 0.33, \rho = 2700\ kg/m^3$)

Figura 17

Longitud de Onda Vs Frecuenca. Ejemplo 2.

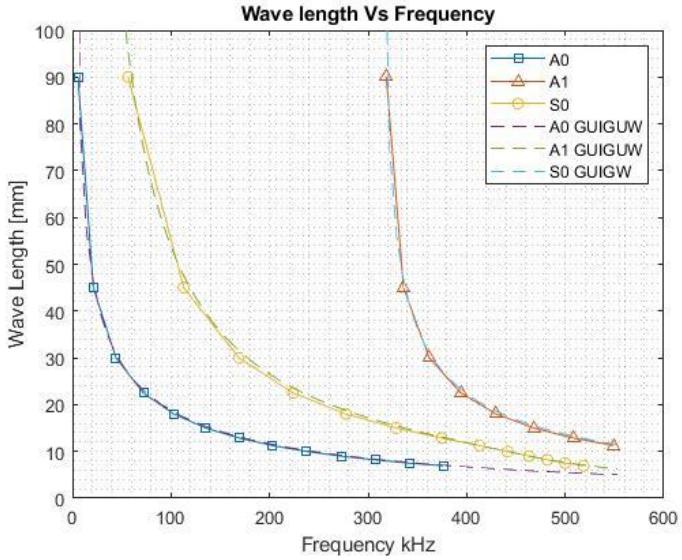


Figura 18

Número de onda Vs Frecuencia. Ejemplo 2.

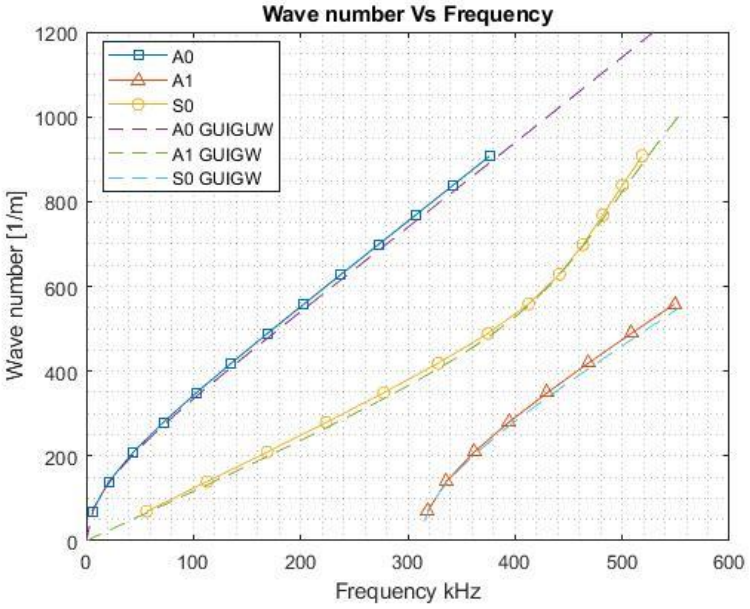


Figura 19

Velocidad de Fase Vs Frecuencia. Ejemplo 2.

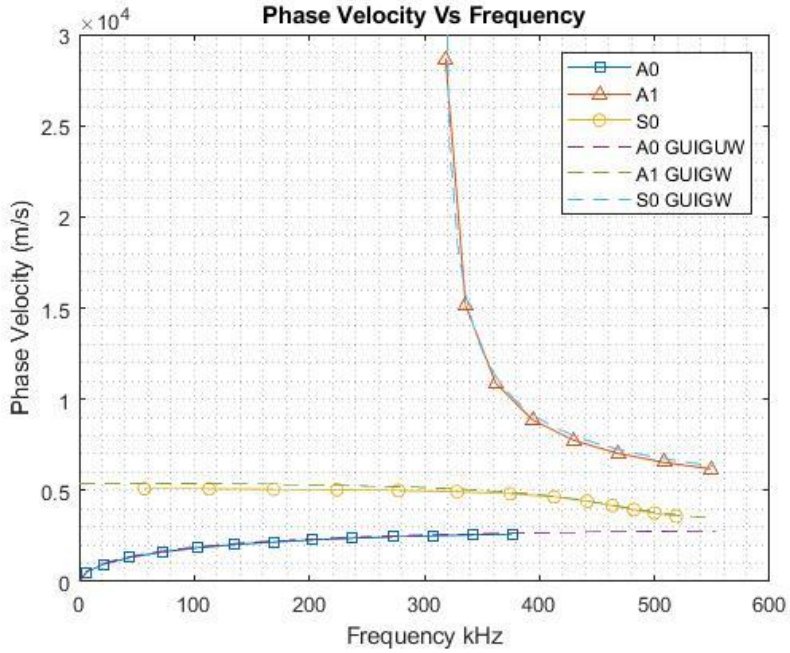
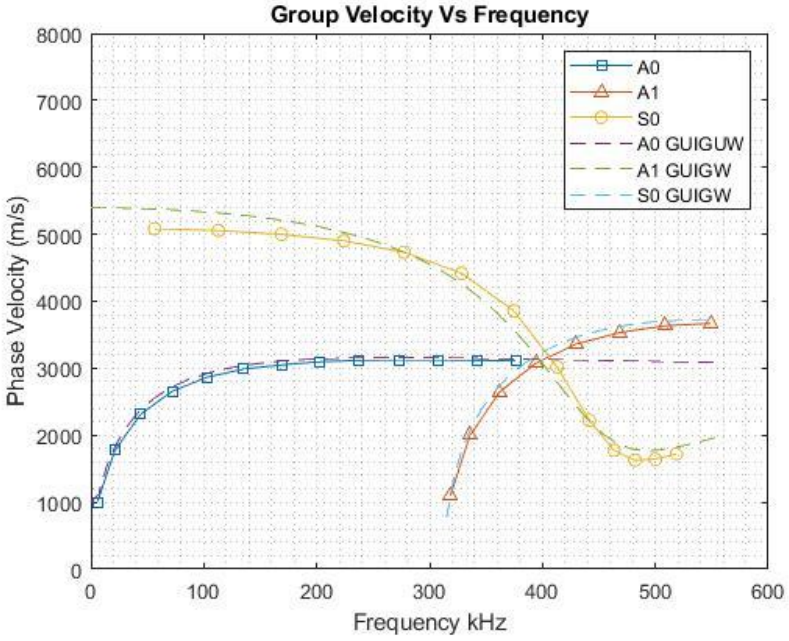


Figura 20

Velocidad de Grupo Vs Frecuencia. Ejemplo 2.



Para el desarrollo de la interfaz gráfica de usuario (GUI), basta con un diseño básico y sencillo, ya que los datos de entrada son solamente los datos de la geometría (altura y longitud) y las propiedades del material (E, v, ρ). La única información de salida son las gráficas, por lo que, para simplificar la GUI, se agrega una lista desplegable en la que el usuario pueda elegir cuál gráfica mostrar en el panel de gráfica. El diseño de la GUI se puede ver en el Apéndice I.

8. Conclusiones

Este trabajo presenta el desarrollo de una aplicación en Matlab para la generación de curvas de dispersión en placas planas isotrópicas empleando la PDE toolbox de Matlab para el desarrollo del modelo de elementos finitos. Dicha aplicación predice frecuencias naturales y modos de vibración con gran exactitud y hace el postprocesamiento de la información dinámica, haciendo que el proceso de cálculo y generación de las curvas de dispersión para placas isotrópicas sea automático, haciendo que no sea necesario un alto grado de conocimiento previo, lo que aumenta la eficiencia del proceso.

Se realizó la validación de los resultados de la aplicación comparando las curvas de dispersión generadas en 2 ejemplos con las generadas por el software GUIGUW. Superponiendo los resultados de las curvas en una misma gráfica se puede analizar gráficamente la correlación entre los 2 resultados, mostrando una similitud aceptable entre las gráficas. Lo que valida el modelo planteado y el desarrollo de la aplicación mediante la metodología establecida.

Se planteó y se desarrolló el modelo de elementos finitos que sirve como base para la generación de curvas de dispersión tanto para tubos y perfiles en I como para cualquier geometría de sección transversal constante. El desarrollo del modelo permite encontrar las frecuencias

naturales y los modos de vibración de la guía de onda, por lo que sirve como base para futuros trabajos relacionados, haciendo falta únicamente el postprocesamiento de la información dinámica.

Se diseñó una interfaz gráfica mediante la cual se facilita el uso de la aplicación desarrollada. Desde esta interfaz el usuario puede generar curvas de dispersión de placas planas de material isotrópico con facilidad. Solamente es necesario ingresar las propiedades del modelo: geometría y propiedades del material.

9. Recomendaciones

El enfoque general de este trabajo es buscar un método de generar curvas de dispersión de manera más eficiente. Los métodos tradicionales de generación de curvas de dispersión hacen que sea indispensable tener un amplio conocimiento del tema para poder interpretar y validar la información que se va obteniendo durante el proceso, a esto se le suma el gran gasto computacional y humano que se debe hacer ya que estos métodos implican la utilización de un software para la solución del modelo de elementos finitos y/o otros para hacer el postprocesamiento de la información dinámica. En este trabajo se desarrolló una aplicación capaz de aumentar la eficiencia del proceso de generación de curvas de dispersión ya que permite la generación de curvas de dispersión sin la necesidad de utilizar otro software y sin necesidad de tener gran conocimiento en el tema.

Adicional al desarrollo de la aplicación para la generación de curvas de dispersión en placas planas, se desarrolló el modelo de elementos finitos necesario para la generación de curvas de dispersión en tubos y perfiles en I. Este modelo de elementos finitos hace el análisis modal de la estructura aplicando las condiciones de contorno periódicas (acople) entre las caras de los extremos

de la geometría. Como recomendación para futuros trabajos, se aconseja evaluar el potencial del método en estructuras de geometría compleja con sección transversal constante. Si bien las condiciones de contorno periódicas representan un gasto computacional mayor que con otras metodologías, presenta un gran potencial frente a las demás. Ya que se puede emplear esta metodología a casi cualquier geometría, dando resultados más precisos y confiables. Partiendo del análisis modal ya desarrollado en este trabajo y buscando la manera de solventar las dificultades presentadas por Matlab en este tipo de cálculos es posible desarrollar una aplicación que sea capaz de generar las curvas de dispersión para cualquier geometría de sección transversal constante.

Referencias Bibliográficas

Auld, B. A., (1975). *Acoustic fields and waves in solid.* New York, Estados Unidos. Wiley-Interscience.

Dunn, M. (2018). *Nonlinear vibro-ultrasonics for detection of damage and wak bonds in composites* (Tesis doctoral). The University of Queensland. Australia.

Grot, E. (2016). *Propagacao de ondas de tensao em hastes retangulares no intervalo de frecuencia de (0;100 [KHz]).* (Tesis de maestría). Universidad Federal Do Rio Grande Do Sul. Brasil.

Idzi, J. (2017). *Estudio numérico de la propagación de ondas guiadas en rieles ferroviarios.* (Tesis de maestría). Universidad Federal Do Rio Grande Do Sul. Brasil.

Lowe, M. J. (1995). Matrix Techniques for Modeling Ultrasonic Waves in Multilayered Media. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, *42*(4), 525–542. https://doi.org/10.1109/58.393096

Partial Differential Equation Toolbox. (Octubre 2020). Mathworks. https://www.mathworks.com/products/pde.html

Rose, J. L. (2003, January). ¨Back to Basics – Dispersion Curves in Guided Wave Testing," Materials Evaluation. *Materials Evaluation, Volume 61, Issue 1*.

Rose, J. L. (2014), *Ultrasonic Guided Waves in Solid Media.* Volumen 1. Cambridge University Press.

Šofer, M., Ferfecki, P., & Šofer, P. (2018). Numerical solution of Rayleigh-Lamb frequency equation for real, imaginary and complex wavenumbers. *MATEC Web of Conferences*, *157*, 08011. https://doi.org/10.1051/matecconf/201815708011

Sorohan, Ș., Constantin, N., Găvan, M., & Anghel, V. (2011). Extraction of dispersion curves for waves propagating in free complex waveguides by standard finite element codes. *Ultrasonics*, *51*(4), 503–515. https://doi.org/10.1016/j.ultras.2010.12.003

Sorohan, Ș., Constantin, N., Găvan, M., & Anghel, V. (2007). Numerical extraction of dispersión curves for Lamb wave inspections on complex structures. https://doi.org/10.1016/j.ultras.2010.12.003

Zhang, W., Hao, H., Wu, J., Li, J., Ma, H., & Li, C. (2018). Detection of minor damage in structures with guided wave signals and nonlinear oscillator. *Measurement: Journal of the International Measurement Confederation*, *122*(March 2018), 532–544. https://doi.org/10.1016/j.measurement.2017.06.033

Zhongqing, S., Lin, Y. & Ye, L. (2006). Guided Lamb waves for identification of damage in composite structures: A review. *Journa of sound and Vibración 3(5), 753-780. https://doi.org/10.1016/j.jsv.2006.01.020*

Apéndices

Apéndice A: Programa principal.

```
%Frequency and wavelength limits(lambda=L/p;lambdamin = L/pmax)
freqMax = 700000;%(Frequency in Hz)
pmax = 14;

%Geometry of plate
len=60; %Length
heigth=4; %Heigth(thick of plate)
len2 = len*1.005;%Perturbation of "delta L" of +0.5%


% Material Properties (Aluminum)
% All properties in mm
E = 70E6;  %Young's Modulus
rho = 2.7E-6;  %Density
nu=0.33; %Poisson's ratio

nmodes = 100;%Number of modes to find in FEA(Calculated modes appear in
pairs)

[modalResults,tmodalResults,Mesh] =
modalAnalysis(len,heigth,E,rho,nu,nmodes);%Modal Analysis 2D with coupling

[NodalDisplacement] = nodalDisplacement(Mesh, modalResults,nmodes);%Nodal
displacements for each mode


[harmonicModes] = harmonicModes(tmodalResults,nmodes);%Classification of
harmonic modes

[SymmetricModes, AntisymmetricModes] =
symmetryMode(harmonicModes,NodalDisplacement,Mesh,heigth);%Classification of
modes Antisymmetric or Symemetric

%Classification of modes by mode-number(p)
%(Classification by wave length)
[pAntisymmetric, pSymmetric] =
modeNumber(Mesh,heigth,AntisymmetricModes,NodalDisplacement,SymmetricModes);

AntisymmetricModes.P = pAntisymmetric; %Object creation with information from
all Symmetric modes

SymmetricModes.P = pSymmetric; %Object creation with information from all
Antisymmetric modes


%Modal Analysis with coupling
```

```
%Length perturbation +0.5%
[modalResults2,tmodalResults2] = modalAnalysis(len2,heigth,E,rho,nu,nmodes);


%Classify Symmetric and Antisymmetric modes in:
%A0,A1,A2,A3,...,An and S0,S1,S2,...,Sn
[clasifTable,casymm,csymm] = clasifTable(AntisymmetricModes,SymmetricModes);

%% Create final Object
%For Symmetric Modes
%Create a final table calculating the wavelength(lambda) and Wave Number(k)
%for each type of Symmetric mode
 for i =1:csymm
    p = 0;
    m = i-1;
    for j = 1:length(clasifTable.("S"+m))
        if clasifTable.("S"+m)(j,3)<=freqMax
            if clasifTable.("S"+m)(j,2)<=pmax
                p = p+1;
                finalTable.("S"+m)(p,1) = clasifTable.("S"+m)(p,2);%p
                finalTable.("S"+m)(p,2) =
clasifTable.("S"+m)(p,3)/1000;%Frequency
%                   finalTable.("S"+m)(p,3) =
tmodalResults2.Frequency(clasifTable.("S"+m)(j,1));%Frequency with +pert
                finalTable.("S"+m)(p,3) = len/p;%Lambda
                finalTable.("S"+m)(p,4) = 2000*pi/finalTable.("S"+m)(p,3);%K
                finalTable.("S"+m)(p,5) =
finalTable.("S"+m)(p,3)*finalTable.("S"+m)(p,2);%Phase Velocity
            end
        end
    end
end

% %For Asymmetric Modes
%Create a final table calculating the wavelength(lambda) and Wave Number(k)
%for each type of Antisymmetric mode
for i =1:casymm
    p = 0;
    m = i-1;
    for j = 1:length(clasifTable.("A"+m))
        if clasifTable.("A"+m)(j,3)<=freqMax %To skip frequencies higher than
the maximum frequency set
            if clasifTable.("A"+m)(j,2)<=pmax %To skip wavelength under the
minimum set
                p = p+1;
                finalTable.("A"+m)(p,1) = clasifTable.("A"+m)(p,2);
                finalTable.("A"+m)(p,2) = clasifTable.("A"+m)(p,3)/1000;
%                   finalTable.("A"+m)(p,3) =
tmodalResults2.Frequency(clasifTable.("A"+m)(j,1));
                finalTable.("A"+m)(p,3) = len/p;%Lambda
                finalTable.("A"+m)(p,4) =
2000*pi/(finalTable.("A"+m)(p,3));%K
                finalTable.("A"+m)(p,5) =
finalTable.("A"+m)(p,3)*finalTable.("A"+m)(p,2);%Phase Velocity
```

```
            end
        end
    end
end

%% Group Velocity

%For Symmetric Modes
for ii = 1:csymm
    i = ii-1;
    for j = 1:length(finalTable.("S"+i))
        groupVelocity.("S"+i)(j,1) = finalTable.("S"+i)(j,1);%P
        %Lambdas
        groupVelocity.("S"+i)(j,2) = len/groupVelocity.("S"+i)(j,1);%Lambda
        groupVelocity.("S"+i)(j,3) =
len2/groupVelocity.("S"+i)(j,1);%Lambda(L+)
        %Wave Number
        groupVelocity.("S"+i)(j,4) = 2*pi/groupVelocity.("S"+i)(j,2);%K(L)
        groupVelocity.("S"+i)(j,5) = 2*pi/groupVelocity.("S"+i)(j,3);%K(L+)
        %Frequencies
        groupVelocity.("S"+i)(j,6) =
tmodalResults.Frequency(clasifTable.("S"+i)(finalTable.("S"+i)(j,1)))/1000;%F
(L)
        groupVelocity.("S"+i)(j,7) =
tmodalResults2.Frequency(clasifTable.("S"+i)(finalTable.("S"+i)(j,1)))/1000;%
F(L-)
        %Deltas
        deltaw = 2*pi*groupVelocity.("S"+i)(j,6)-
2*pi*groupVelocity.("S"+i)(j,7);
        deltak = groupVelocity.("S"+i)(j,4)-groupVelocity.("S"+i)(j,5);
        kf = deltaw/deltak;
        %K in table
        groupVelocity.("S"+i)(j,8) = kf;

    end
end

%For Symmetric Modes
for ii = 1:casymm
    i = ii-1;
    for j = 1:length(finalTable.("A"+i))
        groupVelocity.("A"+i)(j,1) = finalTable.("A"+i)(j,1);%P
        %Lambdas
        groupVelocity.("A"+i)(j,2) = len/groupVelocity.("A"+i)(j,1);%Lambda
        groupVelocity.("A"+i)(j,3) =
len2/groupVelocity.("A"+i)(j,1);%Lambda(L+)
        %Wave Number
        groupVelocity.("A"+i)(j,4) = 2*pi/groupVelocity.("A"+i)(j,2);%K(L)
        groupVelocity.("A"+i)(j,5) = 2*pi/groupVelocity.("A"+i)(j,3);%K(L+)
        %Frequencies
        groupVelocity.("A"+i)(j,6) =
tmodalResults.Frequency(clasifTable.("A"+i)(finalTable.("A"+i)(j,1)))/1000;%F
(L)
```

```
        groupVelocity.("A"+i)(j,7) =
tmodalResults2.Frequency(clasifTable.("A"+i)(finalTable.("A"+i)(j,1)))/1000;%
F(L-)
        %Deltas
        deltaw = 2*pi*groupVelocity.("A"+i)(j,6)-
2*pi*groupVelocity.("A"+i)(j,7);
        deltak = groupVelocity.("A"+i)(j,4)-groupVelocity.("A"+i)(j,5);
        kf = deltaw/deltak;
        %K in table
        groupVelocity.("A"+i)(j,8) = kf;


    end
end

%% To Plot DISPERSION CURVES

%PLOT Wave Length
figure
plot(finalTable.A0(:,2),finalTable.A0(:,3),'-s')
hold on
plot(finalTable.A1(:,2),finalTable.A1(:,3),'-^')
hold on
plot(finalTable.S0(:,2),finalTable.S0(:,3),'-o')
hold off
grid minor
xlabel('Frequency kHz')
ylabel('Wave Length [mm]')
legend('A0','A1','S0')
title('Wave length Vs Frequency')

% Plot Wave number
figure
plot(finalTable.A0(:,2),finalTable.A0(:,4),'-s')
hold on
plot(finalTable.A1(:,2),finalTable.A1(:,4),'-^')
hold on
plot(finalTable.S0(:,2),finalTable.S0(:,4),'-o')
hold off
grid minor
xlabel('Frequency kHz')
ylabel('Wave number [1/m]')
legend('A0','A1','S0')
title('Wave number Vs Frequency')

%Plot Phase Velocity
figure
plot(finalTable.A0(:,2),finalTable.A0(:,5),'-s')
hold on
plot(finalTable.A1(:,2),finalTable.A1(:,5),'-^')
hold on
plot(finalTable.S0(:,2),finalTable.S0(:,5),'-o')
hold off
grid minor
```

```
xlabel('Frequency kHz')
ylabel('Phase Velocity (m/s)')
legend('A0','A1','S0')
title('Phase Velocity Vs Frequency')

%Plot Group Velocity
figure
plot(groupVelocity.A0(:,6),groupVelocity.A0(:,8),'-s')
hold on
plot(groupVelocity.A1(:,6),groupVelocity.A1(:,8),'-^')
hold on
plot(groupVelocity.S0(:,6),groupVelocity.S0(:,8),'-o')
hold off
grid minor
xlabel('Frequency kHz')
ylabel('Group Velocity (km/s)')
legend('A0','A1','S0')
title('Group Velocity Vs Frequency')
```

Apéndice B: Función para el análisis modal

```
%This function dos the 2D modal Analysis with the coupling between the
%nodes of the lateral faces

function [modalResults,tmodalResults,Mesh] =
modalAnalysis(len,heigth,E,rho,nu,nmodes)

% Generic equation based interface, two-PDEs.
model = createpde(2);

le = 0.1; %Element size for Mesh

% Create a 2-D geometry of cross-section

gdm =[3,4,0,len,len,0,0,0,heigth,heigth]';
g = decsg(gdm,'R1',['R1']');
geometryFromEdges(model,g);

% %To Visualize Geometry
% figure
% hc = pdegplot(model,'EdgeLabels','on');
% title('Plate with Edge Labels')

% Define plane-stress c-coefficients
G = E/(2.*(1 + nu));
mu = 2*G*nu/(1 - nu);
c = [2*G + mu; 0; G;   0; G; mu; 0;  G; 0; 2*G + mu];
f = [0 0]';
specifyCoefficients(model,'m',rho,'d',0,'c',c,'a',0,'f',f)

Mesh = generateMesh(model,'Hmax',le)
```

```
% %To visualize Mesh
figure
pdeplot(model,'NodeLabels','off');
% title('Mesh with Quadratic Triangle Elements');

% Extract FE matrices to impose periodic BC. Periodic BC is not supported
% in PDE Toolbox. But it is possible to implement it using dirichlet
constraint
% where ux_boundary1 - ux boundary2 = 0 as constraints.

% Get the discritized matrices to impose the required BCs
feamatWOPeriodicBC = assembleFEMatrices(model);
H = feamatWOPeriodicBC.H;
numNodes = size(model.Mesh.Nodes,2);

%periodic BC pairs
pBC2nodes = findNodes(model.Mesh,'region','Edge',2);
pBC4nodes = fliplr(findNodes(model.Mesh,'region','Edge',4));

% check y-coordinates are same, mesh should match for periodic BC, notice the
fliplr in previous line.
if norm(model.Mesh.Nodes(2,pBC2nodes) - model.Mesh.Nodes(2,pBC4nodes)) > 1E-
12
    error('Mesh on the boundary pair is not compatible to define periodic
BC');
end

[numH, numColsH] = size(H);
numPeriodicCons1 = numel(pBC2nodes); % additional constraints for type
ux_boundary1 - ux boundary2 = 0,
H = [H; zeros(2*numPeriodicCons1, numColsH);]; % Factor 2 is for 2 DoF per
node.
r1 = (numH + 1):(numH + numPeriodicCons1); % Row indices for x-periodic
constraints.
r2 = (numH+numPeriodicCons1+ 1):(numH + 2*numPeriodicCons1); % Row indices
for y-periodic constraints.

%First periodic BC constraints for x-direction
H(r1, pBC2nodes) = eye(numPeriodicCons1);
H(r1, pBC4nodes) = -eye(numPeriodicCons1);
%First periodic BC constraints for y-direction
H(r2, pBC2nodes+numNodes) = eye(numPeriodicCons1);
H(r2, pBC4nodes+numNodes) = -eye(numPeriodicCons1);
% Use update H to impose constraints on stiffness and mass matrices.
[B,~]=pdenullorth(H);
K = feamatWOPeriodicBC.K;
F = feamatWOPeriodicBC.F;
A = feamatWOPeriodicBC.A;
Q = feamatWOPeriodicBC.Q;
G = feamatWOPeriodicBC.G;
M = feamatWOPeriodicBC.M;

KK=K+Q+A;
K=B'*KK*B;
```

```
M = B'*M*B;

% Compute smallest n eigenvalues and vectors
[v,l]=eigs(K,M,nmodes,'sm');
modeID = 1:numel(diag(l));

tmodalResults = table(modeID.',abs(sqrt(diag(l)))/2/pi);
tmodalResults.Properties.VariableNames = {'Mode','Frequency'};
tmodalResults.Frequency = round(tmodalResults.Frequency)
% disp(tmodalResults);

% Add Dirichlet BC DoF back to the solution
modeShapes=B*v;
%Further, it is easier to put the eigenvalues and eigenvectors into a results
objects, which would reshape the data into an easily accessible format to
plot:
% Create an EigenResults object
modalResults = createPDEResults(model,modeShapes,diag(l),'eigen');
end
```

Apéndice C: Función para la creación de una matriz con los desplazamientos nodales.

```
%This function This function creates a multidimensional array in which it
%saves the displacement of each node for each mode calculated by the FEM

function [NodalDisplacement] = nodalDisplacement(Mesh, modalResults,nmodes)

for zz=1:nmodes
    for ii=1:length(Mesh.Nodes)
    %X Displacement
    NodalDisplacement(ii,1,zz) =
Mesh.Nodes(1,ii)+real(modalResults.Eigenvectors(ii,1,zz));
    %Y Displacement
    NodalDisplacement(ii,2,zz) =
Mesh.Nodes(2,ii)+real(modalResults.Eigenvectors(ii,2,zz));
    end
end

end
```

Apéndice D: Función para la clasificación de modos armónicos.

```
%This function eliminates non-harmonic modes calculated by the FEM.
%It's deduced by experimentation that harmonic modes appear in pairs, so
%modes that don't have a pair should be eliminates.

function [harmonicModes] = harmonicModes(tmodalResults,nmodes)

nm = 0;
for i = 1:nmodes-1
    if (tmodalResults.Frequency(i))>0
        if (tmodalResults.Frequency(i)/tmodalResults.Frequency(i+1))>0.99999
            nm = nm+1;
```

```
            harmonicModes.FEAmode(nm,1) = tmodalResults.Mode(i);
            harmonicModes.Frequency(nm,1) = tmodalResults.Frequency(i);
        end
    end
end
harmonicModes = table(harmonicModes.FEAmode,harmonicModes.Frequency);
harmonicModes.Properties.VariableNames = {'FEAmode','Frequency'};

end
```

Apéndice E: Función para la clasificación de modos en Simétricos (S) o Antisimétricos (A)

```
%This function identifies the symmetry of each mode.
%To identify the symmetry of each mode, the waves produced by each mode at
%the top and bottom edge are compared.
%If the behavior of the wave at the top edge is equal to that of the wave
%at the bottom edge, then the mode is Antisymmetric.
%If the behavior of the wave at the top edge is opposite to that of the
%wave at the bottom edge, then the mode is Symmetric.
%The behavior of each mode can be visualized with the plotTB function.

function [SymmetricModes, AntisymmetricModes] =
symmetryMode(harmonicModes,NodalDisplacement,Mesh,heigth,le)

aa = 0;
ss = 0;

le = 0.1; %Element size for Mesh

for i=1:length(harmonicModes.Frequency)
    %d1: 1st node on the TOP EDGE
    %d2: 2nd node on the TOP EDGE
    %d3: 3st node on the BOTTOM EDGE
    %d4: 4st node on the BOTTOM EDGE

    d1 =
NodalDisplacement(findNodes(Mesh,'nearest',([0;heigth])),2,harmonicModes.FEAm
ode(i));
    d2 =
NodalDisplacement(findNodes(Mesh,'nearest',([le/2;heigth])),2,harmonicModes.F
EAmode(i));

    d3 =
NodalDisplacement(findNodes(Mesh,'nearest',([0;0])),2,harmonicModes.FEAmode(i
));
    d4 =
NodalDisplacement(findNodes(Mesh,'nearest',([le/2;0])),2,harmonicModes.FEAmod
e(i));

    if (d1>d2)&&(d3>d4)
        aa = aa+1;
        AntisymmetricModes(aa,1) = harmonicModes.FEAmode(i);
        AntisymmetricModes(aa,2) = harmonicModes.Frequency(i);
```

```
    elseif (d1<d2)&&(d3<d4)
        aa = aa+1;
        AntisymmetricModes(aa,1) = harmonicModes.FEAmode(i);
        AntisymmetricModes(aa,2) = harmonicModes.Frequency(i);
    elseif (d1>d2)&&(d3<d4)
        ss = ss+1;
        SymmetricModes(ss,1) = harmonicModes.FEAmode(i);
        SymmetricModes(ss,2) = harmonicModes.Frequency(i);
    elseif (d1<d2)&&(d3>d4)
        ss =ss+1;
        SymmetricModes(ss,1) = harmonicModes.FEAmode(i);
        SymmetricModes(ss,2) = harmonicModes.Frequency(i);
    end

end

AntisymmetricModes = table(AntisymmetricModes(:,1),AntisymmetricModes(:,2));
AntisymmetricModes.Properties.VariableNames = {'FEAmode','Frequency'};

SymmetricModes = table(SymmetricModes(:,1),SymmetricModes(:,2));
SymmetricModes.Properties.VariableNames = {'FEAmode','Frequency'};

end
```

Apéndice F: Función para la clasificación de modos por el número de modo (p) (Longitud de onda)**.**

```
%This function sorts the modes found by the FEM by "p" number. The
%wavelength is: Lambda = L/p

function [pAntisymmetric, pSymmetric] =
modeNumber(Mesh,heigth,AntisymmetricModes,NodalDisplacement,SymmetricModes)

le = 0.1;

%Find the nodes on a side Edge
nodesTop = findNodes(Mesh,'region','Edge',3);

%Sort the nodes on the top Edge according to its position
for i = 1:length(nodesTop)
    pos = (-le/2)+i*(le/2);
    nodesTopORD(1,i) = findNodes(Mesh,'nearest',([pos;heigth])) ;
end

% For Antisymmetric Modes
% Nodal displacement on top Edge with sorted nodes
for i = 1:length(AntisymmetricModes.Frequency)
    for j = 1:length(nodesTop)
        despTOPAnti(i,j) =
NodalDisplacement(nodesTopORD(1,j),2,AntisymmetricModes.FEAmode(i));
    end
```

```
end

% To determinethe number "p", there must be as many wavelengths are on the
% plate, therefore the nodes and antinodes of the wave must be counted.
% To identify the nodes and anti-nodes, The slope between each displacement
%of each node with the previous one is calculated and compared with the
%previous slope, when there is a change in the sign of the slope and,
%depending on the change, it is classified as a node or antinode
%(the change from a positive to negative slope is an antinode.
%The change from negative to positive slope is a node)

%For Antisymmetric Modes
for i = 1:length(AntisymmetricModes.Frequency)
    mant = 0;
    nodos = 0;
    antinodos = 0;
    for j = 2:length(nodesTop)
        m = despTOPAnti(i,j)-despTOPAnti(i,j-1);
        if (mant*m)<0
            if m>mant
                nodos = nodos + 1;
            else
                antinodos = antinodos + 1;
            end
        end
        mant = m;
        pAntisymmetric(i,1) = nodos;

    end
end



% For Symmetric Modes
for i = 1:length(SymmetricModes.Frequency)
    for j = 1:length(nodesTop)
        despTOPSymm(i,j) =
NodalDisplacement(nodesTopORD(1,j),2,SymmetricModes.FEAmode(i));
    end
end

for i = 1:length(SymmetricModes.Frequency)
    mant = 0;
    nodos = 0;
    antinodos = 0;
    for j = 2:length(nodesTop)
        m = despTOPSymm(i,j)-despTOPSymm(i,j-1);
        if (mant*m)<0
            if m>mant
                nodos = nodos + 1;
            else
                antinodos = antinodos + 1;
            end
        end
```

```
        mant = m;
        pSymmetric(i,1) = nodos;

    end
end


end
```

Apéndice G: Función para la clasificación de los modos antisimétricos en A0, A1, etc. Y los modos simétricos en S0, S1, etc.

```
%This function classifies Symmetric and Antisymmetric modes in
%(A0,A1,A2,...,An) and (S0,S1,S2,...,Sn)

function [clasifTable,casymm,csymm] =
clasifTable(AntisymmetricModes,SymmetricModes)

%For Antisymmetric Modes
casymm = 0;
for i = 1:length(AntisymmetricModes.P)
    if AntisymmetricModes.P(i) == 1
        casymm = casymm+1;
    end
end

for i =1:casymm
    f = 0;
    for j = 1:length(AntisymmetricModes.P)
    p = 0;
        for k = 1:j
            if AntisymmetricModes.P(j) == AntisymmetricModes.P(k)
                p = p+1;
            end
        end
        if p == i
            f = f+1;
            m = i-1;
            clasifTable.("A"+m)(f,1) = AntisymmetricModes.FEAmode(j);
            clasifTable.("A"+m)(f,2) = AntisymmetricModes.P(j);
            clasifTable.("A"+m)(f,3) = AntisymmetricModes.Frequency(j);
        end
    end
end



%For Symmetric Modes
```

```
csymm = 0;
for i = 1:length(SymmetricModes.P)
    if SymmetricModes.P(i) == 1
        csymm = csymm+1;
    end
end

for i =1:csymm
    f = 0;
    for j = 1:length(SymmetricModes.P)
    p = 0;
        for k = 1:j
            if SymmetricModes.P(j) == SymmetricModes.P(k)
                p = p+1;
            end
        end
        if p == i
            f = f+1;
            m = i-1;
            clasifTable.("S"+m)(f,1) = SymmetricModes.FEAmode(j);
            clasifTable.("S"+m)(f,2) = SymmetricModes.P(j);
            clasifTable.("S"+m)(f,3) = SymmetricModes.Frequency(j);
        end
    end
end



end
```

Apéndice H: Programa para el análisis modal con condiciones de contorno periódicas en 3 DOFs (para geometrías 3D)

```
% This program does modal analysis with periodic boundary conditions with 3
DOFs for 3D geometries.
model = createpde('structural','modal-solid');

nmodes=100;

%Geometry
Rext = 7;
th = 1;
Rin = Rext-th;
heigth = 15;

gm = multicylinder([Rin Rext],heigth,'Void',[true,false]);
model.Geometry = gm;
pdegplot(model,'CellLabels','on','FaceAlpha',0.5,'FaceLabels','on')

%Material Properties
%E = 212E6;
```

```
%nu = 0.29;
%rho = 7.85E-6;
E = 212e6;
nu = .29;
rho = 7.850e-6;
structuralProperties(model,'YoungsModulus',E,'PoissonsRatio',nu,'MassDensity'
,rho);

%Mesh
Mesh = generateMesh(model,'Hmax',1); %,'GeometricOrder','linear');
pdeplot3D(model)

%the variable "FEM" is now a struct containig the finite element matrices
FEM = assembleFEMatrices(model);

%Nodes in Faces
pBC1nodes = findNodes(model.Mesh,'region','Face',1);
pBC1Coords = model.Mesh.Nodes(:,pBC1nodes);
aligned_pBC2Coords = pBC1Coords +[0;0;heigth];
pBC2nodes_aligned = findNodes(model.Mesh,'nearest',aligned_pBC2Coords);
pBC2nodes = pBC2nodes_aligned;
%Add one row for each pair of node points to "H" and "R"
H = FEM.H;
numNodes = size(model.Mesh.Nodes,2);
[numH,numColsH] = size(H);
numPeriodicCons1 = numel(pBC1nodes);
%H = [H; zeros(3*numPeriodicCons1, numColsH);];
numH = 0;
H = zeros(3*numPeriodicCons1, numColsH);
r1 = (numH + 1):(numH + numPeriodicCons1); % Row indices for x-periodic
constraints.
r2 = (numH+numPeriodicCons1+ 1):(numH + 2*numPeriodicCons1); % Row indices
for y-periodic constraints.
r3 = (numH+2*numPeriodicCons1+1):(numH + 3*numPeriodicCons1);% Row indices
for z-periodic constraints.

%First periodic BC constraints for x-direction
H(r1, pBC1nodes) = eye(numPeriodicCons1);
H(r1, pBC2nodes) = -eye(numPeriodicCons1);
%First periodic BC constraints for y-direction
H(r2, pBC1nodes+numNodes) = eye(numPeriodicCons1);
H(r2, pBC2nodes+numNodes) = -eye(numPeriodicCons1);
%First periodic BC constraints for z-direction
H(r3, pBC1nodes+2*numNodes) = eye(numPeriodicCons1);
H(r3, pBC2nodes+2*numNodes) = -eye(numPeriodicCons1);


% Use update H to impose constraints on stiffness and mass matrices.
[B,~]=pdenullorth(H);
K = FEM.K;
F = FEM.F;
A = FEM.A;
Q = FEM.Q;
G = FEM.G;
```

```
M = FEM.M;

KK=K+Q+A;
K=B'*KK*B;
M = B'*M*B;

%Compute smallest n eigenvalues and vectors
[v,l]=eigs(K,M,nmodes,'sm');
modeID = 1:numel(diag(l));
tmodalResults = table(modeID.',abs(sqrt(diag(l)))/2/pi);
tmodalResults.Properties.VariableNames = {'Mode','Frequency'};
disp(tmodalResults);

modeShapes=B*v;
```

Apéndice I: Diseño de la interfaz gráfica de usuario (*GUI)*