Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Mecánica

Desarrollo de un software libre para un entorno web de análisis modal de sistemas discretos unidireccionales

Autor:

Llorente Pascual, Rodrigo

Tutor:

Magdaleno González, Álvaro

Departamento de Construcciones Arquitectónicas, Ingeniería del Terreno y Mecánica de los Medios Continuos y Teoría de Estructuras

Valladolid, septiembre 2021.

Agradecimientos

A mi tutor y el personal del departamento, por ayudarme siempre y enseñarme todo lo necesario para poder finalizar este trabajo con el que cierro muy satisfecho la carrera como Ingeniero Mecánico.

A mi familia y amigos por estar siempre apoyándome en todo y creyendo en mí desde el principio.

RESUMEN

Este trabajo de fin de grado tiene como finalidad el desarrollo de un programa basado en el lenguaje de programación de Python para procesar los datos de un sistema amortiguado y estudiar su comportamiento tanto en el tiempo como su respuesta en frecuencia. El programa también dispondrá de una interfaz para interactuar y para la visualización de los datos basada en la librería Bokeh del mismo lenguaje, Python. Se ha programado para un entorno web y que en un futuro se pueda acceder desde un servidor en la universidad.

Palabras clave: Python, Respuesta en el tiempo y Frecuencia, Bokeh

ABSTRACT:

The purpose of this final degree project is the development of a software based on Python Scripting language, which is able to process data of a damped system and study its response, in both the time and the frequency domains. The software will also include an interface in order to interact and visualize data, which is based on the library Bokeh of the same Python language. It has been created on a web environment in order to enable in the future free access from a server at the university.

Key words: Python, Time and frecuency response, Bokeh

Contenido

TABLA DE FIGURAS

Capítulo I. Introducción, objetivos y planteamiento

1.1 Introducción

Gracias a los avances en el ámbito de las técnicas de construcción y de los materiales empleados en las últimas décadas, las estructuras han evolucionado hacia diseños cada vez más complicados. Por ello son necesarios métodos de cálculo más complejos para procesar los datos que se tienen sobre ellas y así comprender el comportamiento de esas estructuras frente a cargas dinámicas, todo esto con la finalidad y propósito de desarrollar un modelo.

Un modelo es una representación simplificada de la estructura que se quiere estudiar, esto se hace porque la realidad es demasiado compleja para analizarla. Existen dos tipos de modelos posibles, modelos experimentales como puede ser la recreación de una estructura, pero con simplificaciones en cuanto a detalles y variación del tamaño y modelos computacionales basados en programas de cálculo matemático. En este trabajo se centra en el último modelo, el computacional, pues la finalidad es desarrollar un programa que pueda estudiar diferentes sistemas, aunque es necesario el uso de modelos experimentales para la validación del mismo.

Como condiciones a cumplir, un modelo debe ser muy parecido a la estructura real y a su vez, sencillo. Cuanto más se parezcan el modelo a la estructura mejor serán los resultados, aunque esto, conlleva mayor complejidad en los cálculos. Es necesario conocer el nivel de similitud del modelo y la estructura pues de esto dependerá en que el modelo sea válido y los resultados exactos. Una vez se analizan los datos obtenidos tras ensayar el modelo bajo unas condiciones similares a las de la estructura real, se obtienen los parámetros modales, estos son los que describen las propiedades dinámicas de la estructura. Además, los datos obtenidos se contrastan con parámetros fiables con el fin de refinar el modelo y obtener resultados más exactos.

Es importante destacar la normativa, ésta exige garantías de seguridad y servicio que tradicionalmente se evalúan mediante análisis estáticos. Estos análisis son insuficientes en casos con estructuras muy esbeltas, pues ante efectos dinámicos pueden afectar enormemente a su funcionalidad. Efectos como terremotos o la vibración que generan los peatones en un puente, influyen directamente en el confort y por lo tanto en la salud de los usuarios de esas estructuras.

En cuanto a los ensayos experimentales, son caros, difíciles de ser ejecutados e interpretados y además llevan tiempo. Estos necesitan ser reproducidos computacionalmente, por lo que hace falta un modelo que permitan simplificar el análisis y pre dimensionar la estructura incluso antes de empezar la construcción. De esta manera los modelos son muy útiles a la hora de predecir la vida a fatiga y las respuestas en resonancia, estos interpretan las propiedades dinámicas propias de una estructura.

Los modelos ayudan a tener una descripción de las estructuras, pudiendo usarla para evaluar las especificaciones que tendrá en el diseño y en el análisis de estructuras ya existentes. No solo se habla de construcciones, también se hacen modelos de estructuras como automóviles o maquinaria que experimenta vibraciones, siendo un posible problema para su desempeño. Los modelos alcanzan casos sencillos o enormemente complejos, en función del uso que se les dé, pues dependerá de lo más o menos avanzado que se encuentre el proceso de diseño.

En este sentido, los modelos más complejos dan una visión más acertada de la realidad y por lo tanto más precisa mientras que los modelos sencillos son fáciles de manipular y desarrollar, pero se alejan más de la realidad. En esto, los modelos sencillos siguen siendo muy útiles porque, al fin y al cabo, son suficientes para evaluar y comprender cómo puede responder una estructura.

1.2 Objetivos

El objetivo de este Trabajo Fin de Grado es desarrollar un programa orientado como un código abierto que permita analizar las propiedades dinámicas y modales de un sistema discreto compuesto por masas puntuales, resortes y amortiguadores ideales. Para darle forma al código, es necesario una interfaz, pues el usuario tiene que ser capaz de poder introducir todos los datos y propiedades de la estructura y obtener resultados y gráficas tras analizarlo. Es decir, el usuario tiene que ser capaz de poder usarlo sin necesidad de entender el código que tiene detrás.

Con todo esto, que sea accesible es fundamental. Por ello, se programa de forma que esté listo para lanzarlo desde un servidor y se pueda usar de manera online. Una vez se ha finalizado el código es necesario realizar las comprobaciones y análisis del software para verificar que todo funciona correctamente y solucionar problemas de la puesta en marcha. Problemas como pueden ser la adaptación del tamaño de la interfaz al mayor número de pantallas o el correcto funcionamiento y acceso desde el servidor.

1.3 Planteamiento

En términos generales la parte teórica y el desarrollo del software se divide en dos partes diferentes. La primera parte es el estudio del sistema o estructura en el tiempo frente a fuerzas externas, pudiendo estudiar las posiciones, velocidades y aceleraciones de los diferentes grados de libertad. Esto es útil para aproximar el comportamiento que tendrá una estructura frente a unas fuerzas exteriores conocidas o por lo contrario determinar esas fuerzas a partir de los datos del sistema como el tiempo y los desplazamientos. La segunda rama se centra en el estudio de respuesta en frecuencia sobre el sistema, siendo este quizá más importante que el anterior, aunque también más complejo porque en dinámica de estructuras es necesario resolver problemas de valores propios a partir de las matrices físicas.

Se ha programado en el lenguaje de programación Python, que destaca por su legibilidad disponiendo de una gran cantidad de librerías para el cálculo matemático que lo hacen ideal para el motor de nuestro programa. En este proyecto destacan la librería Numpy [1] o Scipy [2] que son muy usadas en el ámbito científico. Se ha planteado de tal forma que sea un código principal el que llame a las diferentes funciones que se han creado, tanto para el aparato matemático como para la parte de la interfaz. Se ha dispuesto de esta manera con el fin de optimizar el proceso y de que la distribución sea más cómoda y legible para alguien que lo interprete desde fuera. Siguiendo esta lógica, de haber un problema con un error en el código también es más fácil de localizar y solucionar.

Entre estas funciones a las que llama el código principal estará la que corresponde a la interfaz, para esta parte se ha usado la librería Bokeh [3] que se ha elegido entre otras como Dash [4] por llevar más tiempo y tener una comunidad más amplia detrás. Esto es importante a la hora de encontrar soluciones o problemas muy específicos ya que con su trayectoria es fácil de encontrar sitios como los propios foros de ayuda de Bokeh donde la comunidad comparte alternativas u opciones a multitud de problemas comunes o muy específicos en un campo concreto y todo ello actualizado.

Para la interfaz se ha tenido en cuenta que el programa será de acceso remoto por un servidor local en la universidad por lo que se tiene que adaptar a las diferentes pantallas que usaran el programa. Es importante tener en cuenta la disposición o el layout de la interfaz del programa, donde las diferentes gráficas y apartados donde pide información al usuario deben ser intuitivos y claros de usar.

Figura 1.1 : Logo de Python y Bokeh

Capítulo II. Marco Teórico

2.1 Introducción

El análisis modal se rige por las ecuaciones de movimiento, que dependen de las características del sistema como la masa, la rigidez y el amortiguamiento. Para la resolución se impone una fuerza que es dependiente del tiempo. Una vez se resuelve el sistema, se obtienen los desplazamientos, velocidades o aceleraciones de los grados de libertad en función del tiempo. Para la parte de análisis modal como tal se supone una forma de respuesta y se impone que cumpla las ecuaciones que rigen el movimiento del sistema, esto conlleva resolver un problema de valores propios.

Para un grado de libertad, la ecuación es sencilla de resolver, pero en la realidad los sistemas son muy diferente y el número de grados de libertad puede ser mayor. Por esto, es necesario crear un sistema de grado “n”, siendo “n” los grados de libertad del sistema, facilitando así la resolución del problema.

El análisis modal permite detallar y describir la estructura por sus propiedades dinámicas como la masa y la rigidez además de los parámetros modales como la frecuencia. Todas las estructuras tienen frecuencias naturales y modos de vibración que permite comprender y analizar el comportamiento de las estructuras.

2.2 Ecuaciones de un modelo discreto unidireccional

El objetivo principal de la parte teórica empieza con definir cuál va a ser el modelo que se va a estudiar, en este Trabajo Fin de Grado será un modelo discreto unidireccional de n grados de libertad. Para explicar los conceptos que involucran a este modelo, lo más sencillo, es reducirlo a un solo grado de libertad, que es el sistema que se muestra en la Figura 2.1.

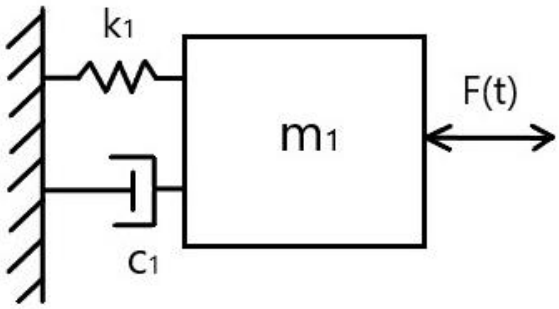


Figura 2.1 : Esquema de un sistema de un grado de libertad

Este esquema muestra un sistema de un grado de libertad con una masa m, un muelle con una constante elástica k, un amortiguador con un coeficiente de amortiguamiento c y una fuerza de excitación F(t) que es dependiente del tiempo. La masa tiene una posición de equilibrio que se tiene en cuenta a la hora de calcular sus amplitudes. Para un grado de libertad, siendo w la coordenada, la ecuación que rige el movimiento, Ecuación 2.1, sale del sumatorio de fuerzas del sistema, Ecuación 2.2 y Ecuación 2.3.

$$m\,\ddot{w} + c\,\dot{w} + k\,w = F \tag{2.1}$$

$$-c\,\dot{w} - k\,w + F = m\,\ddot{w} \tag{2.2}$$

$$F_c + F_k + F = m\,\ddot{w} \tag{2.3}$$

Ahora es necesario plantear un sistema, para n grados de libertad, que pueda resolver los diferentes sistemas. El esquema que representa el sistema genérico sería el siguiente que se muestra en la Figura 2.2.

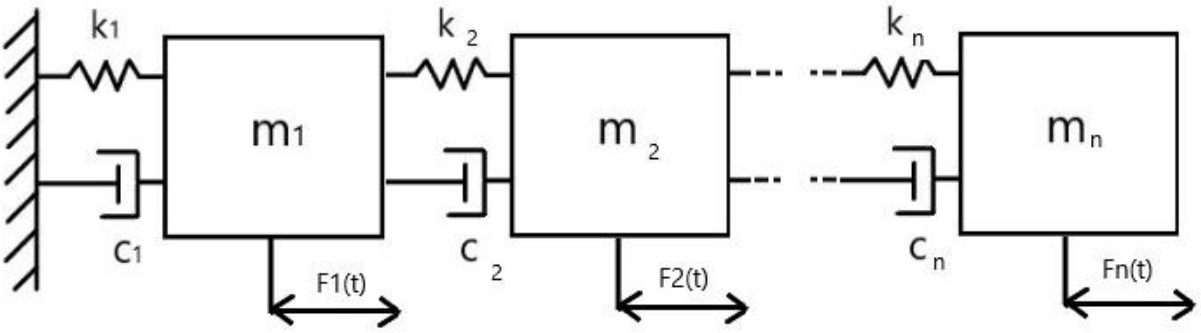


Figura 2.2: Esquema de un sistema de n grados de libertad

A diferencia de antes, cada grado de libertad depende de sus propiedades y de la fuerza que se le aplica. Las ecuaciones que rigen ahora el sistema se vuelven más sofisticadas. Se ven en el Sistema de Ecuaciones 2.4 y 2.5 como sería para n grados de libertad con w como coordenadas.

$$\begin{cases} F_{C1} + F_{K1} - F_{C2} - F_{K2} + F_1 = m_1 \cdot \ddot{w}_1 \\ F_{C2} + F_{K2} - F_{C3} - F_{K3} + F_2 = m_2 \cdot \ddot{w}_2 \\ \cdots \\ F_{Cn} + F_{Kn} + F_n = m_n \cdot \ddot{w}_n \end{cases} \tag{2.4}$$

$$\begin{cases} m_1\ddot{w}_1 + (c_1 + c_2)\dot{w}_1 - c_2\dot{w}_2 + (k_1 + k_2)w_1 - k_2w_2 = F_1 \\ m_2\ddot{w}_2 - c_2\dot{w}_1 + (c_2 + c_3)\dot{w}_2 - c_3\dot{w}_3 - k_2w_1 + (k_2 + k_3)w_2 - k_3w_3 = F_2 \\ \cdots \\ m_i\ddot{w}_i - c_i\dot{w}_{i-1} + (c_i + c_{i+1})\dot{w}_i - c_{i+1}\dot{w}_{i+1} - k_iw_{i-1} + (k_i + k_{i-1})w_i - k_{i+1}w_{i+1} = F_i \end{cases} \tag{2.5}$$

Como se ve, se crea un patrón de n ecuaciones dinámicas con n incógnitas (w_1, w_2, …, w_i, …, w_n). Aprovechando esto se puede escribir de forma más compacta a través de las matrices de masa(M), amortiguamiento (C) y rigidez (K), de forma que se obtiene la Ecuación 2.6.

$$M\ddot{q} + C\dot{q} + Kq = F \tag{2.6}$$

Donde q es un vector columna con los grados de libertad como se ve en la Ecuación 2.7.

$$q = \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_i \\ \cdots \\ w_n \end{bmatrix} \tag{2.7}$$

Por lo que las matrices toman la forma de las Ecuaciones 2.8, 2.9 y 2.10.

$$M = \begin{bmatrix} m_1 & & & & & \\ & m_2 & & & & \\ & & \ddots & & & \\ & & & m_i & & \\ & & & & \ddots & \\ & & & & & m_n \end{bmatrix} \tag{2.8}$$

$$C = \begin{bmatrix} c_1 + c_2 & -c_2 & & \cdots & & \\ -c_2 & c_2 + c_3 & & & & \\ & & \ddots & -c_i & & \vdots \\ & & -c_i & c_i + c_{i+1} & & \\ \vdots & & & & \ddots & -c_n \\ & & & \cdots & -c_n & c_n \end{bmatrix} \tag{2.9}$$

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & & \cdots & & \\ -k_2 & k_2 + k_3 & & & & \\ & & \ddots & -k_i & & \vdots \\ & & -k_i & k_i + k_{i+1} & & \\ \vdots & & & & \ddots & -k_n \\ & & & \cdots & -k_n & k_n \end{bmatrix} \tag{2.10}$$

2.3 Resolución del sistema mediante un espacio de estados

Con la Ecuación 2.6 queda definido el sistema de n grados de libertad, la resolución del sistema es más complicado, pues es de orden dos y su solución no es inmediata. Se reduce mediante el Espacio de Estados empezando por definir un vector de estado que contiene desplazamientos y velocidades, Ecuación 2.11. La Ecuación 2.12 representa la derivada del vector de estado, que tiene velocidades y aceleraciones.

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \tag{2.11}$$

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} \tag{2.12}$$

De esta forma la Ecuación 2.6 pasa a ser la Ecuación 2.13.

$$\ddot{q} = -M^{-1}C\dot{q} - M^{-1}\,Kq + M^{-1}F \tag{2.13}$$

Haciendo uso del vector de estado se puede reescribir la ecuación anterior como el Sistema 2.14, siendo las Ecuaciones de estado.

$$\begin{bmatrix}\dot{q}\\ \ddot{q}\end{bmatrix} = \begin{bmatrix}0 & I_n\\ -M^{-1}K & -M^{-1}C\end{bmatrix}\begin{bmatrix}q\\ \dot{q}\end{bmatrix} + \begin{bmatrix}0\\ M^{-1}\end{bmatrix}[F] \tag{2.14}$$

Abreviadamente se puede expresar como la Ecuación 2.15. Donde las matrices A y B son las Ecuaciones de entrada y un vector u es la matriz de Fuerzas.

$$\dot{x} = Ax + Bu \tag{2.15}$$

Estas ecuaciones se complementan con la Ecuación 2.16, denominándose Ecuaciones de salida. Esta C' no es la misma C que la de la Ecuación 2.9, una es la matriz de amortiguamiento (C) y la otra es la matriz de salida de la formulación del espacio de estados. Es necesario remarcarlo para evitar confusiones.

$$y = C'x + Du \tag{2.16}$$

El vector ***y*** es el vector de salida, y contiene las magnitudes que se quieren sacar como respuesta del modelo (desplazamiento, velocidades, aceleraciones, reacciones, etc.). Siendo estas las salidas que buscará el usuario del programa.

Las matrices tomaran una forma u otra, en el caso de los desplazamientos o velocidades es tan sencillo como colocar unos (1) en la matriz C de forma que "seleccione" los elementos. Un ejemplo es si en un sistema de tres grados de libertad, que tendrá tres desplazamientos y tres velocidades, se quiere sacar solo los deslazamientos, escribiendo la Ecuación 2.16 como la Ecuación 2.16.1.

$$y = \begin{bmatrix}w_1\\ w_2\\ w_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}w_1\\ w_2\\ w_3\\ \dot{w}_1\\ \dot{w}_2\\ \dot{w}_3\end{bmatrix} + 0\cdot u \tag{2.16.1}$$

También, es necesario poder hacer combinaciones lineales, por ejemplo, si se quiere la combinación lineal de los dos primeros desplazamientos tal que $2w_1 - 3w_2$ y la suma de las dos primeras velocidades, entonces la Ecuación 2.16 toma la forma de la Ecuación 2.16.2.

$$y = \begin{bmatrix} 2w_1 - 3w_2 \\ \dot{w}_1 + \dot{w}_2 \end{bmatrix} = \begin{bmatrix} 2 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \end{bmatrix} + 0 \cdot u \qquad (2.16.2)$$

Ahora, el caso de las aceleraciones es diferente al de los desplazamientos y velocidades. Solo se complican un poco más que las anteriores, aunque se simplifica ya que la expresión que las define está ya escrita en la parte inferior de las Ecuaciones de Estado 2.14. Si se quieren todas las aceleraciones, algo habitual, se coge la mitad inferior de la matriz A como matriz C y toda la mitad inferior de la matriz B como matriz D, la cual ya no es nula como antes, quedando la Ecuación 2.17.

$$y = \ddot{q} = [-M^{-1}K \quad -M^{-1}C] \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + M^{-1}F \qquad (2.17)$$

En el caso de que se necesite solo una o unas pocas aceleraciones, bastaría con seleccionar la o las filas correspondientes de estas últimas matrices.

Este sería el procedimiento a seguir para plantear el sistema de Ecuaciones de estado y el cual será el que se implemente en el código de programación más adelante. El usuario del programa será el que escoja las salidas, modificando el programa las matrices correspondientes. Todo esto tiene el propósito de hacer más sencilla la resolución del problema planteado.

2.4 Problema de valores propios

En análisis modal la forma de proceder es plantear la ecuación de movimiento del sistema, visto en el Apartado 2.2, a esta ecuación se le impone que cumpla una forma de respuesta, lo que conduce a un problema de valores propios. En mecánica de vibraciones, o dinámica de estructuras en este caso, se pueden resolver dos tipos de problemas de valores propios a partir de las matrices físicas anteriores, el no amortiguado y el amortiguado.

Los valores propios dan información sobre las distintas formas modales asociadas a las n frecuencias propias. Estas son las distintas formas en las que oscila la estructura cuando alcanza la resonancia al ser excitada con una fuerza alternante a la correspondiente frecuencia propia, estos valores son útiles pues ayudan en el estudio dinámico del sistema además de que se pueden graficar para su interpretación.

2.4.1 Problema de valores propios del caso no amortiguado

Al no contar con el amortiguamiento la matriz C es nula y la ecuación a resolver en este apartado es la Ecuación 2.18.

$$(K - \omega^2 M)v_r = 0 \tag{2.18}$$

Para operar la ecuación se usa una función en concreto de la librería Scipy de Python, scipy.linalg.eig. Después de que se hayan calculado los autovalores (ω) se comprueba que sean valores reales pues puede haber quedado algún residuo imaginario en el cálculo. Además, se transforman a frecuencias propias de radianes por segundo (rad/s) a hercios (Hz) dividiendo por 2π.

Para el caso de los auto vectores (v_r), el resultado es una matriz cuadrada de dimensión n. Cada columna de esta matriz es una forma modal asociada a cada una de las n frecuencias propias que se han calculado arriba.

2.4.2 Problema de valores propios del caso amortiguado

Al incluir la matriz C la ecuación evoluciona de la vista en la Ecuación 2.18 a la que se muestra ahora, la Ecuación 2.19.

$$({s_r}^2 M + s_r C + K)v_c = 0 \tag{2.19}$$

Siendo la ecuación de segundo grado es necesario reducirla como se hizo para el espacio de estados. Se parte del sistema de ecuaciones visto anteriormente, la Ecuación 2.20.

$$M\ddot{q}(t) + C\dot{q}(t) + Kq(t) = F(t) \tag{2.20}$$

Siendo este de segundo orden es necesario hacer el cambio de variable $x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$ para transformarlo al siguiente sistema de primer orden, la Ecuación 2.21.

$$A\dot{x}(t) + Bx(t) = u_f(t) \tag{2.21}$$

Ahora las matrices A, B y u_f toman la siguiente forma:

$$A = \begin{bmatrix} C & M \\ M & 0_n \end{bmatrix} \quad ; \quad B = \begin{bmatrix} K & 0_n \\ 0_n & -M \end{bmatrix} \quad ; \quad u_f(t) = \begin{bmatrix} F(t) \\ 0 \end{bmatrix}$$

Siendo ahora una ecuación de primer orden se puede resolver “directamente”. Se usará la misma función vista antes, scipy.linalg.eig (a,b) [5], para sacar los valores propios. En este caso a=B y b=A siendo parecida a la Ecuación 2.18 pero para el caso amortiguado, quedando la Ecuación 2.22:

$$(S_r A + B)v_{rc} = 0 \tag{2.22}$$

Una vez se opera, se obtienen los autovalores (S_r), valores complejos, y autovectores (v_c) para el caso amortiguado. A diferencia del caso no amortiguado, estos valores propios no van a ser del tamaño “n” (números de grados de libertad), sino “2n”. Esta diferencia de dimensiones se debe a operar con matrices de tamaño “2n”, pero tanto los autovalores como los autovectores tienen pares conjugados, es decir, de entre los valores solo hay “n” distintos siendo su módulo la frecuencia propia en rad/s. Al trabajar con Hz, como antes, se divide por 2π. La forma de los autovalores (S_r) es compleja, calculándose como se indica en la Ecuación 2.23. Los factores de amortiguamiento critico (dr), son los valores de los respectivos amortiguamientos para los diferentes grados de libertad que anulan la vibración, siendo el límite entre que la estructura oscile o no. Se obtienen rápidamente de la siguiente forma, siendo la que se ha implementado en el código:

$$s_r = -\omega_r d_r + j\ \omega_r\sqrt{1 - d_r^2} \tag{2.23}$$

$$\left(\frac{-(Parte\ real\ autovalores)}{Módulo\ autovalores}\right) = -\frac{Re(s_r)}{|s_r|} = dr \tag{2.24}$$

Para los autovectores, igual que con los autovalores, los números también aparecen en pares conjugados. En cambio, a diferencia que el anterior, aparecen el doble de componentes, esto es porque por la manera en la que se calcula se tiene a la vez las formas modales correspondientes a los desplazamientos, siendo las “n” (números de grados de libertad) primeras componentes, y las velocidades siendo las siguientes “n” componentes. Escogiendo las “n” primeras componentes de cada vector y solo los vectores diferentes, queda por fin una matriz cuadrada de dimensión n compleja.

2.5 Funciones de respuesta en frecuencia

Las funciones de respuesta en frecuencia o F.R.F. son funciones de transferencia que permite evaluar en uno o más nodos (grados de libertad) la respuesta en frecuencia con una fuerza de excitación unitaria aplicada en un nodo. Estas se pueden graficar, para ello es necesario resolver el sistema que rige el modelo, pero a diferencia del apartado 2.3 solo se buscará una salida pues para las funciones de respuesta en frecuencia se actúa en un nodo y se observa desde otro.

Por lo tanto, será necesario volver a resolver el sistema que rige el modelo mediante un espacio de estados, pero con algunas variantes de lo ya visto. La estructura del sistema no cambia siendo la misma que las vistas en las Ecuaciones 2.25 y 2.26.

$$\dot{x} = Ax + Bu \tag{2.25}$$

$$y = Cx + Du \tag{2.26}$$

Las matrices de entrada (A y B) se quedan prácticamente iguales, cambiando solo la matriz B, pues al querer solo una salida se multiplica a la parte inferior de la matriz B por un vector de dimensión n donde tiene un solo uno como amplitud en la posición sobre la que actuará la fuerza, como se muestra en la Figura 2.2 para el caso con tres grados de libertad y actuando sobre el segundo.

$$B = \begin{bmatrix} 0 \\ M^{-1}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix} \tag{2.27}$$

Para las matrices de salida (C y D), la matriz C se compone solo de unos en la primera diagonal como se ve en la Ecuación 2.28, pues solo interesan los desplazamientos para sacar las receptancias, para la movilidad (velocidades) y acelerancias (aceleraciones) se sacan a partir de las receptancias. Por esto mismo la matriz de salida D será nula, ya que no se buscan las aceleraciones.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.28}$$

Una vez se resuelve el sistema con el mismo procedimiento visto en el apartado 2.3, ya se obtienen la receptancias y se pueden graficar frente a la frecuencia para sacar las funciones de respuesta en frecuencia, siendo los valores picos de estas gráficas la frecuencia natural de la estructura.

Finalmente, para obtener las movilidades a partir de las receptancias se multiplica por la unidad imaginaria y por la frecuencia. De la misma manera, para obtener acelerancias se puede volver a multiplicar por la unidad imaginaria y por la frecuencia de nuevo.

Capítulo III. Desarrollo del motor de cálculo en el software

3.1 Introducción

El motor de cálculo es la parte más importante, aunque no siempre la que se ve, pues un usuario no tiene por qué saber nada del código que tiene detrás para entender el programa. Esto no quita que el código no tenga que estar bien estructurado y sea legible pues, como se mencionaba antes, será un software de código abierto y tiene que ser fácil de interpretar para alguien que lo ve desde fuera. Una de las ventajas de que sea de código abierto y esté bien estructurado es la posibilidad de que alguien en el futuro pueda trabajar en él, tanto para ampliarlo como para usarlo de referencia en un programa similar.

El código se ha estructurado tal que sea un archivo principal el que llame a las diferentes funciones que se han creado, agilizando así el código y permitiendo recurrir más adelante a estas funciones sin compilar todo de nuevo. Esto es importante tenerlo en cuenta desde el principio pues según se desarrolla el programa es más difícil de cambiar y es necesario que esté preparado para funcionar rápido y no perder tiempo de compilación en líneas de código innecesarias. Además, el que este estructurado por funciones diferentes permite localizar un problema más rápido ya que en esos casos se busca el problema en la función que falla y no en todas las líneas de código.

En general el código contará con tres partes diferentes, el código principal que se mencionaba y dos partes que se dividen en las funciones de cálculo modal, el primero, y el que lleva la parte de la interfaz gráfica, el segundo. El código principal tiene la función de introducir los datos con los que empieza a funcionar el código, además de llamar a las funciones que se encargarán del cálculo modal y la interfaz gráfica.

La parte del código sobre el cálculo modal se centra en el estudio en el tiempo y análisis modal, este último se encarga de la parte matemática y teórica vista en el Capítulo 2 de este trabajo. En esta parte del código es importante la agilidad de cálculo y como se han resuelto los diferentes problemas, cuantas menos líneas de código se

usen para realizar la misma tarea menos tiempo necesita el programa para ejecutarse. Esto influye en la rapidez con la que el programa reacciona a diferentes entradas, por esto es importante estudiar las funciones que ofrece Bokeh [3].

Por último, la parte del programa que trata sobre la interfaz gráfica, basada en la librería Bokeh mencionada antes, y que se ha diseñado pensando en el usuario para que le sea lo más intuitiva, cómoda y útil posible la aplicación. Para esta parte es importante analizar que hace a un programa intuitivo y fácil de usar, por lo que es necesario estudiar cómo están diseñados los programas similares del mismo ámbito. Destaca en la mayoría una distribución en la interfaz gráfica donde existe una parte que contiene los comandos de entrada de datos o inputs y una parte que contiene una ventana de salida de información, outputs.

3.2 Librerías usadas de Python

Para agilizar el código se usan diferentes librerías que ya están preparadas para ciertas funciones como pueden ser operadores matemáticos. Las librerías de Python que se han usado en este trabajo son:

- Numpy: Es una librería especializada en el cálculo numérico y el análisis de grandes cantidades de datos [1].

- Scipy: Esta es una librería científica que incluye otras herramientas más específicas como Matplotlib [6], Pandas [7] y SymPy [8] juntándolos todos en una sola.

- Matplotlib: Es una librería especializada en la representación de gráficos en dos dimensiones. Esta librería solo se ha usado durante el desarrollo del código para la comprobación de las gráficas [6].

- Bokeh: Esta ha sido clave en el desarrollo de la interfaz gráfica. Una librería especializada en la representación de gráficos y tableros interactivos creada para el diseño de webs.

3.3 Simulación en el dominio del tiempo

Esta parte corresponde a la primera mitad del bloque de cálculo que tiene el código y que está basada en los fundamentos teóricos vistos en el Capítulo 2, Apartado 2.3, sobre la resolución mediante un espacio de estados. Se explica cómo se ha procedido para adaptarlo al código y como se han ido resolviendo los problemas que han aparecido.

3.3.1 Matrices de masa, amortiguamiento y rigidez

Son las respectivas matrices M, C y K vistas en el Apartado 2.3 de este trabajo. Se crean a partir de los datos iniciales que se han introducido como datos para el cálculo o inputs en el código principal. La parte del código, en la que están estas matrices denominadas M la de masas, C la de amortiguamiento y K la de los muelles, será la función: Matrices (m, c, k). Como inputs se llaman a las matrices que contienen los datos sobre el sistema m, c y k, y como respuesta la función devuelve las matrices M, C y K. En la Figura 3.1 se muestra como se define la función y la orden que devuelve los valores de las matrices en el código.

```
def Matrices(m,c,k):
```

```
return M,C,K
```

Figura 3.1: Función Matrices (m, c, k) y comando return

Dentro de la función lo primero que se ha hecho ha sido crear las matrices vacías del tamaño que corresponde. Esto es así porque más adelante las operaciones que se han usado para rellenarlas con sus datos necesitaban una base sobre la que trabajar. Para no sobrecargar la función con demasiados parámetros a la hora de llamarla, el número de grados de libertad n se ha extraído del tamaño de la matriz m, esto es una técnica que se usará más adelante con el fin de simplificar y agilizar el código. Para las matrices vacías se ha usado la función numpy.zeros [9], como se muestra en la Figura 3.2.

```
#Creamos las matrices del tamaño deseado
n = m.shape[0] #GDL
M = np.zeros((n,n))
C = np.zeros((n,n))
K = np.zeros((n,n))
```

Figura 3.2: Creación de matrices vacías

Para completar las matrices se ha procedido de la siguiente forma: para la matriz M se ha recurrido a la función numpy.diag [10] la cual completa la matriz que deseemos con la de dentro de los paréntesis en los huecos de la diagonal de la misma. Para la matriz C y K se ha estudiado el patrón que siguen en los sistemas de n grados de libertad y se ha dispuesto en un bucle que las completa en un ciclo de n vueltas. De querer cambiar algún dato se volvería a llamar a la función con las nuevas matrices de entrada y recalcularía las de salida.

3.3.2 Espacio de Estados

Esta función es la encargada de procesar los datos y la parte del Espacio de Estados vista en el punto 2.3. Como parámetros a los que hace referencia la función están las tres matrices de masa, rigidez y amortiguamiento, vistas en el punto anterior, y las matrices de salida que el usuario solicita.

La función se llama: mat2ss(M, C, K, Posic, Veloc, Acel), da como salida un parámetro que representa al sistema ya simplificado con el espacio de estados, el parámetro es el resultado de la función: signal.StateSpace(As , Bs , Cs , Ds) [11], en el código lleva el nombre de “sys”. Como se ve, la función que crea el sistema con el espacio de estados necesita las matrices de entada A y B de la Ecuación 2.25 y las matrices de salida de la Ecuación 2.26, todas del Apartado 2.3. Como ya se ha visto más arriba la manera en la que se construyen, se explica solo el procedimiento que se ha seguido para crearlas.

Para la matriz de entrada A, se necesitan unir cuatro matrices diferentes en una sola, siendo una de estas una matriz unidad para la que se ha usado la función numpy.eye(n) [12] siendo n la dimensión de la matriz cuadrada. Para unir diferentes “arrays” y que el resultado final siga siendo de tipo “array” se han usado la función numpy.concatenate((a, b), axis=1) [13] siendo a y b las dos matrices a unir y axis=1 si se quiere unirlas en fila o axis=0 en columna. Para la matriz B se ha procedido igual que en la A, con la diferencia de que en el caso de la inversa de la matriz se ha usado la función linalg.inv(M) [14].

Para las matrices de salida (C' y D) existe el problema de que no parten de la unión de matrices cuadradas por lo que es necesario determinar el tamaño de filas, el de columnas es el número de grados de libertad para D y el doble para C', como se había visto. Así, se implementa un contador que da ese valor de filas para poder crearlas vacías y completarlas más adelante según las salidas que busque el usuario.

```
#Determinamos el tamaño de filas de Cs:
Tam = 0
for a in range(0,n):
   Sum = Posic[a]+Veloc[a]+Acel[a]
   Tam = Tam+Sum

Cs =  np.zeros((Tam,n*2))
Ds = np.zeros((Tam,n))
```

Figura 3.3: Creación de las matrices de salida C y D

Para finalizar, queda completar estas matrices de salida que se hacen con bucles ya que permite adaptarse a diferentes valores de grados de libertad, a excepción de la parte que corresponde con las aceleraciones que cambia ligeramente, pues se completa como se mencionaba en el Apartado 2.3.

```
for a in range(0,n): #Aceleraciones
    if Acel[a] == 1:
        Cs[Cont][:] = As2[a][:]
        Ds[Cont][:] = Bs[a+n][:]
        Cont=Cont+1
```

Figura 3.4: Ejemplo de bucle para completar ecuaciones de salida

Por último, queda llamar a la función como: sys = signal.StateSpace(A, B , C , D). Siendo "sys" el sistema con el espacio de estados que devuelve la función de este apartado como retorno.

3.3.3 Simulación del sistema lineal en el tiempo

En este bloque del código, la función tiene la tarea de simular el sistema antes visto con el espacio de estados, en el tiempo. La función se denomina simT(sys,F,dT,T,ws,Desf), necesitando llamar a los parámetros como el sistema que simulará (sys), los parámetros de los grados de libertad de la ecuación de fuerza siendo las amplitudes (F), las frecuencias (ws) y desfases (Desf), siendo estos dos últimos en radianes, y por último a los parámetros que controlan el tiempo de simulación como el tiempo máximo que simulará en segundos y el diferencial de tiempo que es la inversa de la frecuencia de muestreo. Dentro del tipo de fuerzas se ha programado para introducir en el programa fuerzas senoidales sencillas, esto es porque son las más comunes, lo habitual es encontrarse fuerzas alternantes que pueden descomponerse en otras diferentes.

Para simular la salida de un sistema lineal de tiempo continuo se usará la función signal.lsim(sys, Fs, Ts) [14]. Ésta necesita tres parámetros para funcionar, siendo el primero el sistema, segundo el vector de fuerzas que actúa en el sistema y tercero el vector del tiempo del que depende el de fuerzas. El tamaño tanto del vector de fuerzas y del de tiempo es esa dimensión resultante de dividir el tiempo de simulación entre el diferencial, siendo los pasos de tiempo en los que se define la entrada y en los que se desea la salida. El vector Ts de tiempo se crea con la función numpy.arange(0 , T , dT) [15] necesitando el inicio, el final y el diferencial de tiempo. El vector de fuerzas se completa con un bucle que depende del tamaño del vector de tiempo, como se ve en la Figura 3.5.

```
Ts = np.arange( 0 ,  T , dT)

num_elem = Ts.shape[0]
Fs = np.zeros((num_elem,n))
for a in range(0,n):#Fuerzas
    Fs [:,a]= F[a]*np.sin(ws[a]*Ts+Desf[a])
```

Figura 3.5: Vector de Fuerzas en la simulación del sistema

Como se ve en la figura, cada columna del vector de fuerzas “Fs” hace referencia a cada uno de los grados de libertad. La finalidad del programa es poder graficar en función del tiempo estos resultados como se ve en la Figura 3.6 obtenida del programa.

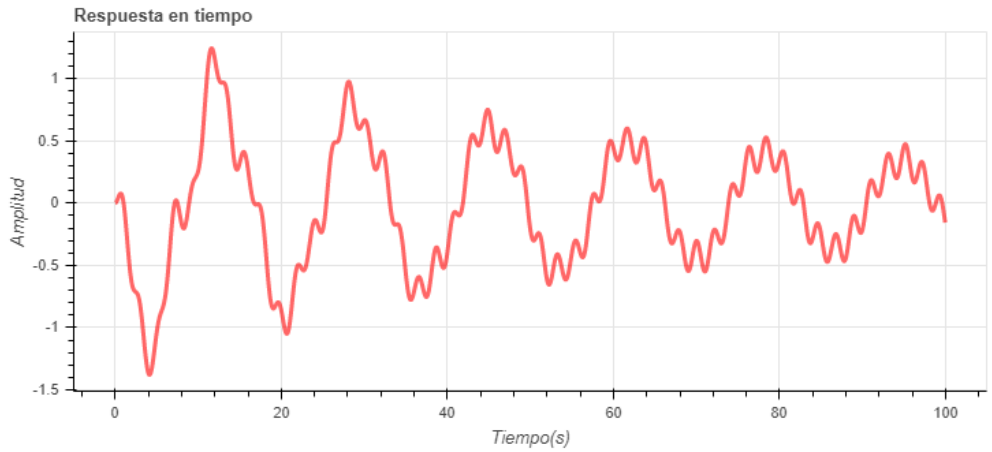


Figura 3.6: Ejemplo de una gráfica de simulación del sistema en el tiempo

Las “irregularidades” que puede presentar la gráfica frente a una onda sinodal perfecta se debe a la superposición de los distintos modos de vibración del sistema. Siendo un modo de vibración el desplazamiento relativo de un nodo respecto del resto, este no es un valor en concreto, por eso su manera de interpretarlo es graficando cada modo de forma independiente. El resultado de superponer varios modos da como resultado los diferentes picos que aparecen en una sola onda de la gráfica, como se ve en la Figura 3.6.

3.4 Estudio en frecuencia del sistema

Esta es la segunda mitad del bloque de cálculo que tiene el código y que está basada en los fundamentos teóricos vistos en el Apartado 2.4. Se explica cómo se ha procedido para adaptarlo al código y como se han ido resolviendo los diferentes problemas. El estudio del análisis en frecuencia tiene como finalidad el cálculo de las frecuencias naturales de las vibraciones de la estructura, esto es para analizar la respuesta dinámica del sistema bajo cargas variables. Aquí es importante el fenómeno de resonancia.

La resonancia es la tendencia de un edificio a oscilar en sus amplitudes más grandes en ciertas frecuencias. El fenómeno de resonancia es importante tenerle en cuenta durante la fase de diseño y cálculo de una estructura, pese a su poca posibilidad de ocurrir, pues es un problema derivado de las vibraciones que puede llevar a una estructura o sistema a colapsar.

3.4.1 Problemas de valores propios

Una vez visto el análisis del sistema frente a fuerzas en el tiempo, toca la parte de cálculo modal. Esta parte se centra en los problemas de valores propios, siendo el nombre de la función: calc_modal(M, C, K). Esta solo necesita la información sobre las propiedades dinámicas del sistema pues no le hace falta más. Como respuesta, en el "return" de la función, devolverá los valores propios tanto del caso no amortiguado como del amortiguado además de los factores de amortiguamiento critico en el caso amortiguado.

- **Caso no amortiguado:**

  Para el caso no amortiguado basta con usar la función scipy.linalg.eig() [5], que se mencionaba en el apartado de problemas de valores propios. Esta función ya resuelve el sistema que se veía en el apartado 2.4.1, en la Ecuación 2.18 sobre valores propios, dando como entrada los datos de la matriz de masas M y la de rigidez K, como salida por un lado los autovalores al cuadrado (w^2) y por otro los autovectores (vr). En el caso de los autovalores se les elimina una posible parte imaginaria que haya podido quedar como residuo, esto es porque no son números exactos y además el propio programa necesita redondear y se pierden datos. Se hace con el propio soporte de Python para números complejos, tal que el parámetro seguido de ". real" devuelve ya esa parte real. Además, se pasa de "w^2" a "w" con la función numpy.sqrt [16] y se transforma de radianes por segundo (rad/seg) a hercios (Hz) dividiendo por 2π.

- **Caso amortiguado:**

En este caso la cosa se complica más, como se ve en el apartado teórico 2.4.2, siendo el mismo sistema que se ha seguido antes para reducir el orden del sistema (Ecuación 2.19) mediante un cambio de variable. Para ello se construyen las matrices de entrada A y B como se muestran en la Ecuación 2.20 y con la misma función vista antes numpy.concatenate(( ) , axis = 1) [13]. Una vez están listas se llama a la función que da los valores propios dando como entrada esas matrices A y B y devolviendo como salida los autovalores con su parte imaginaria (sr) y los autovectores (V_c).

Como se había visto en el apartado teórico, estos valores propios no van a ser del tamaño "n" (siendo n el número de grados de libertad), sino "2·n" y para los autovectores "2n" componentes en vez de "n" componentes por lo que es necesario quitar las partes de valores similares y quedarse solo con los valores únicos. Para los autovalores se sacan los módulos, pues esto resulta ser las frecuencias propias en radianes por segundo (rad/seg) y dividiéndolo por 2π pasan a hercios (Hz). Para los factores de amortiguamiento critico se usa la Ecuación 2.22.

En los autovectores, solo es necesario escoger las tres primeras componentes de cada vector y solo los vectores diferentes, queda por fin una matriz cuadrada de dimensión n compleja, por esos se recurre a la función numpy.ix_ [17], permite seleccionar las filas y columnas con las que quedarse y esta selección se hace con las funciones de Python "list" y "range" como se muestra en la Figura 3.7.

```
 #Autovectores
x = list(range(n))
y = list(range(0,2*n,2))
vc = q[np.ix_(x,y)]
```

Figura 3.7: Selección de los autovectores en el caso amortiguado

3.4.2 Cálculo de las funciones de respuesta en frecuencia.

Esta parte del código corresponde al apartado 2.5 del bloque teórico, la función tiene el nombre de simF(M, C, K, v_act, wt), como se ve, necesita las matrices que definen las propiedades dinámicas de la estructura (M, C, K) además del parámetro que indica en que nodo se aplicará la fuerza (v_act) y el vector de frecuencia (wt).

La forma en la que se ha desarrollado el código es similar a la del apartado 3.3.2 sobre la resolución del sistema mediante un espacio de estados, a diferencia de variar ciertas matrices según se veía en el apartado 2.5. Por otro lado, la forma en la que se

almacenan estos datos también es diferente pues se varía la matriz de salida C para dar las diferentes salidas en función de los diferentes nodos y almacenarlos en una matriz por columnas, siendo una columna por salida o siendo lo mismo, por grado de libertad. Esta parte del código sobre las salidas es la que se ve en la Figura 3.8.

```
for a in range(0,n):
    C = np.zeros((1,2*n))
    C[0][a] = 1
    ss = signal.StateSpace( A , B , C , D )
    wt,H1 = signal.freqresp(ss,wt)
    H[:,a] = H1[:]
```

Figura 3.2: Almacenamiento de las salidas en las Receptancias

Como resultado esta función devuelve un vector de tipo complejo, por tratar con números imaginarios, con n columnas. El resultado de graficar la magnitud de una de esas columnas frente al vector de frecuencias es la función de respuesta en frecuencia del nodo al que corresponda la columna, como en el ejemplo de la Figura 3.9.

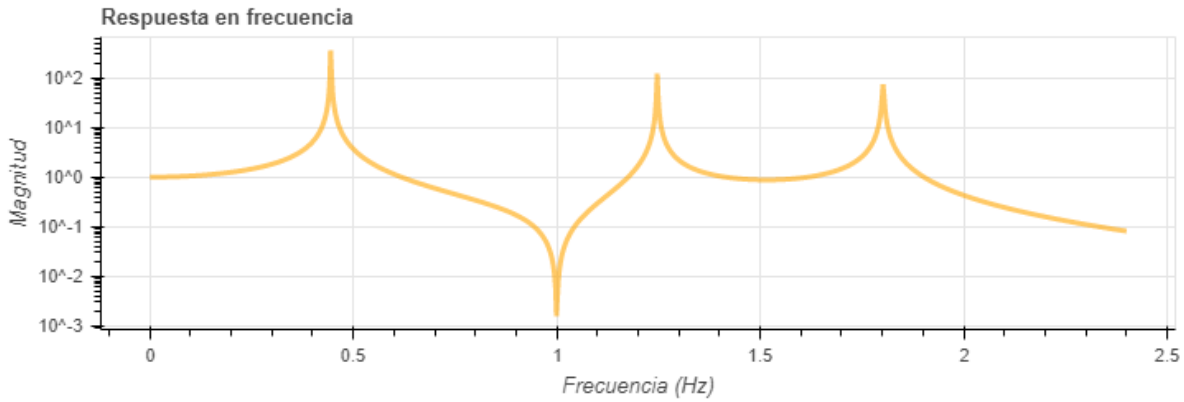


Figura 3.3: Ejemplo de una función de respuesta en frecuencia

En el caso de la Figura 3.9 es de un sistema de tres grados de libertad, donde se aprecian tres picos diferentes de amplitudes máximas siendo las frecuencias propias del sistema

Capítulo IV. Desarrollo de la Interfaz Gráfica del software

4.1 Introducción

Terminado el motor de cálculo del programa, empieza la parte sobre la interfaz gráfica. Es importante en cuanto que es lo que el usuario ve desde fuera, por esto debe ser fácil de interpretar y usar. Esta parte se desarrolla en la librería Bokeh [3] de Python, siendo una librería abierta de uso gratuito con una gran comunidad detrás, esto es útil para buscar soluciones a problemas comunes y funciones en las que la gente ya ha trabajado. La librería tiene como particularidad su capacidad para crear diagramas interactivos listos para su uso en la web.

Uno de los factores más importantes a tener en cuenta es el tamaño de las pantallas, como se desarrolla con el fin de poder ser usado de forma remota, la variedad de dispositivos con los que se puede acceder es grande y el tamaño de pantallas más, por lo que se estima un tamaño compatible para pantallas con una resolución de 1280x720, siendo un tamaño muy común.

4.2 Disposición y layout

Antes de empezar con el desarrollo de la interfaz se necesita una idea o un esquema de cómo será la distribución de los elementos en la pantalla, aquí es necesario pensar en la comodidad y la forma más fácil para que el usuario lo entienda e interprete. La idea principal distribuye el espacio de trabajo en dos mitades o columnas, como se ha indicado en la Figura 4.1, la primera de la izquierda siendo la encargada de la gestión de inputs o entradas, donde el usuario introduzca los datos y los varíe según el estudio. La segunda columna, sería la encargada de la salida de la información por pantalla como las gráficas o las tablas con datos.

En la primera columna están las herramientas más importantes que controlan el programa, siendo la primera un contador de números enteros para el numero de grados de libertad. El segundo elemento es una tabla que muestra los datos de las

masas, rigideces y amortiguamientos, justo debajo de esta está el botón para actualizar estos valores de la tabla por si se modifican. El elemento del medio es un contador que permite elegir sobre que grados de libertad del sistema se quiere observar para sacar la información por pantalla. En las siguientes pestañas son opciones generales para controlar los parámetros del estudio en el tiempo o en frecuencia, así como el valor del tiempo o el rango de frecuencias sobre el que graficar. El último elemento de esta primera columna, es el botón de descarga de datos sobre los parámetros que se han configurado y que se muestran en ese momento por pantalla.

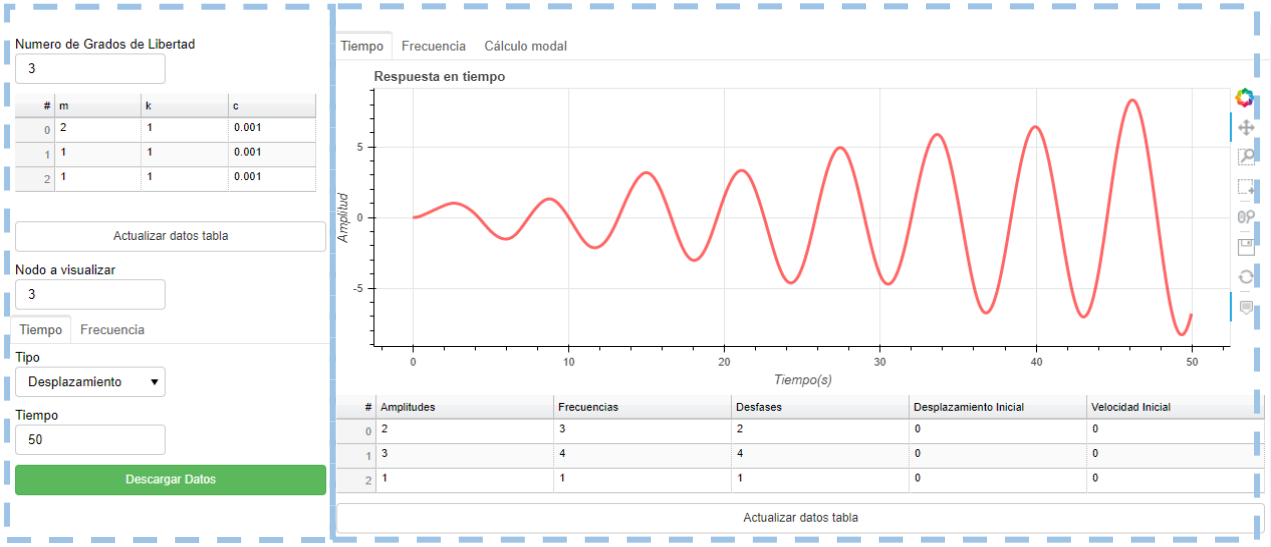


Figura 4.1: Distribucion del layout para la respuesta temporal

Puesto que no es posible sacar todas las salidas por pantalla, la columna de la izquierda cambia por completo dependiendo de si el usuario quiere la respuesta temporal, el análisis modal o los valores de parámetros como los autovectores, autovalores y los factores de amortiguamiento crítico. De esta forma se empaqueta cada apartado por ventanas diferentes a las que se puede acceder a través de las pestañas superiores que se ven en la Figura 4.1.

En la Figura 4.1, debajo de la gráfica que representa el desplazamiento frente al tiempo, hay una excepción a la distribución principal. Es una tabla donde se pueden modificar valores e introducir nuevos sobre las fuerzas senoidales y las condiciones iniciales. Se ha incluido en este apartado por no tener relación con el resto.

Ahora, sobre el estudio en frecuencia, se disponen dos gráficas diferentes. La primera grafica es sobre las funciones de respuesta en frecuencia o F.R.F. que representan la amplitud frente a la frecuencia. Ésta proporciona datos importantes como pueden ser

las frecuencias naturales del sistema. La segunda grafica muestra la fase de los números complejos de las funciones de respuesta en frecuencia frente a la frecuencia.

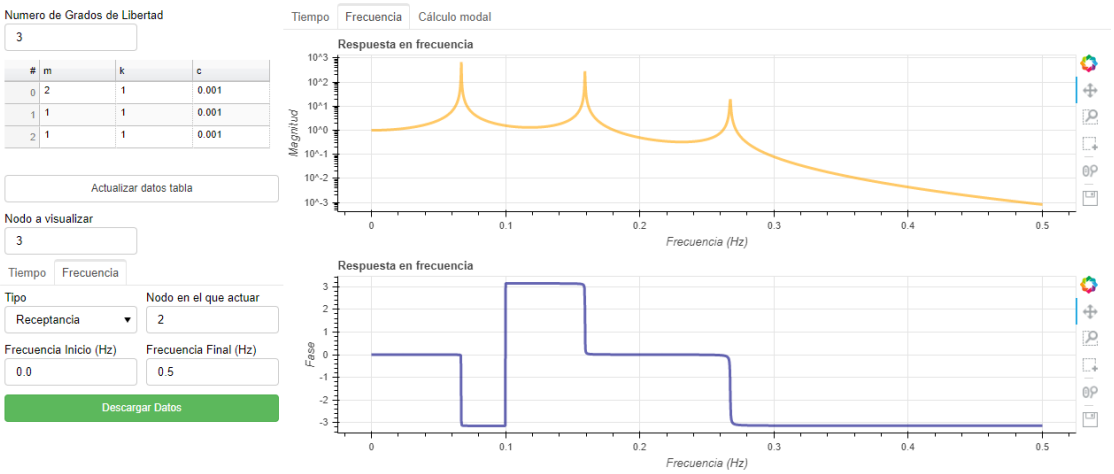


Figura 4.2: Distribucion del layout para la respuesta en frecuencia

La última ventana, proporciona los valores de los autovectores y autovalores en los casos amortiguado y no amortiguado además de los factores de amortiguamiento crítico. En el caso no amortiguado los resultados son números complejos y por las limitaciones de la librería de Bokeh se han distribuido en dos tablas diferentes, las partes reales e imaginarias, como se muestra en la Figura 4.3.

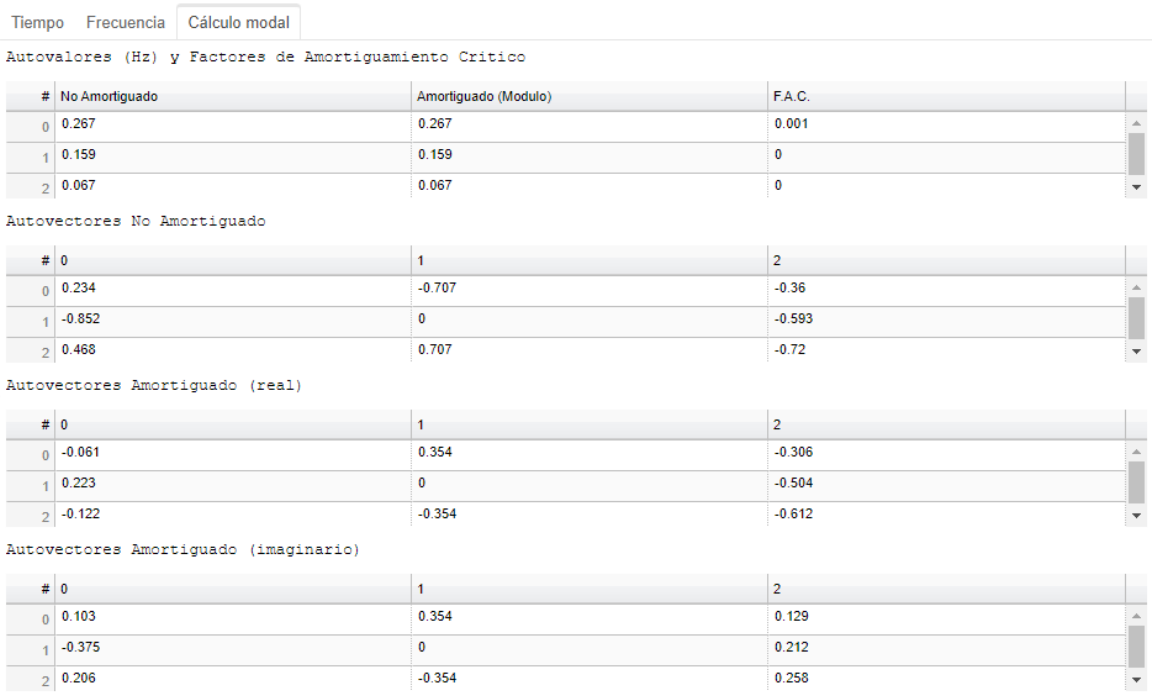


Figura 4.3: Distribucion del layout para los autovalores, autovectores y FAC.

4.3 Herramientas

Este apartado del código se centra en los elementos de entrada de los datos por pantalla, los inputs. La librería de Bokeh dispone de multitud de elementos para añadir al layout como todos los que se ven en la columna de la izquierda, los usados en este proyecto son los que se mencionan a continuación.

- Spinners

  Para la entrada de números enteros, que en este caso está relacionada con los grados de libertad, permite añadir o resta con unas flechas para evitar el error de añadir elementos con decimales que el programa no entienda. Estos elementos dejan escoger el paso de número.

- Select

  Este tipo de herramientas son ventanas con desplegables que muestran diferentes opciones de una lista, se ha usado en los casos como la selección de salida por pantalla como desplazamiento, velocidades o aceleraciones.

- TextInput

  Para la entrada de números que pueden ser decimales los spinner queda descartados, para ellos se eligen las entradas de texto normal, aunque en este caso se han escogido solo para la entrada de números.

- Button

  Estos elementos son simple botones a los que se les puede dar diferentes funciones, en el código en especial tienen las funciones de actualizar cierta entrada de datos o la opción de descarga de información en un archivo externo.

Cada tipo de herramientas tiene ciertas características que se pueden determinar, pero en común tienen elementos que se pueden modificar en todos como el nombre o título, el valor inicial que puede ser numérico o un texto y el tamaño que tendrá esa herramienta en el espacio de la pantalla. Todo esto se puede ver en la Figura 4.4

```
tipo = Select(title="Tipo", value="Velocidad",width=150,
              options=['Desplazamiento', 'Velocidad', 'Aceleracion'])
```

Figura 4.4: Ejemplo de programación de una herramienta tipo Select.

4.4 Graficas Interactivas

Unas de las claves por las que Bokeh destaca frente a otras librerías es la opción de crear diagramas interactivos con multitud de diseños que se pueden encontrar en la página oficial de Bokeh, siendo todos de código abierto. Entre las opciones en las que estos diagramas son interactivos hay multitud de herramientas que se pueden añadir como por ejemplo la opción de descargar las gráficas, hacer recortes de ciertas zonas o moverlas libremente en el espacio designado, en la Figura 4.5 se ven las usadas en el programa y en las Figuras 4.1 y 4.2, a la derecha de las gráficas, se ven las diferentes herramientas que se pueden activar y usar.

```
tools = "save,reset,pan,wheel_zoom,box_zoom,box_select,hover"
```

Figura 4.5:Ejemplo de las herramientas de diagramas en bokeh .

Entre las funciones importantes en este apartado destacan tres que son fundamentales. La primara es la función 'ColumnDataSource' [18], en la parte de diagramas de Bokeh es necesario que los datos que se van a representar estén en una cadena de caracteres que la librería pueda interpretar.

```
source1 = ColumnDataSource(data=dict(x=T_out, y=y1))
```

Figura 4.6: Ejemplo de la función ColumnDataSource de bokeh.

La segunda función importante en esta parte es la del diseño de las gráficas, en concreto la de las curvas, la función 'line'. Como se mencionaba, Bokeh es amplio en cuanto al tema de los diseños de las gráficas, con esta función se puede escoger desde el grosor de las líneas hasta los colores y la forma de representar los puntos que unen las líneas.

```
color = np.array(["navy","green","orange","red","teal","grey","yellow","olive","purple"])
p1.line('x', 'y',source=source1, line_width=3, line_alpha=0.6,color=color[3])
```

Figura 4.7: Ejemplo de la función Line de bokeh.

Por último, la función 'figure', esta es la que genera las gráficas en sí usando las funciones antes mencionadas. Además, al igual que las herramientas, tiene ciertos parámetros que se pueden dejar por defecto como el título de la gráfica o el de los ejes y el tamaño de la misma, este último es importante para la distribución.

```
p1 = figure(title="Respuesta en tiempo",plot_height=336, plot_width=925, tools=tools,
           x_axis_label='Tiempo(s)',y_axis_label='Amplitud')
```

Figura 4.8: Ejemplo de la función Figure de bokeh.

4.5 Tablas de datos

En el desarrollo de la interfaz las tablas tienen dos funciones diferentes, la de entrada y salida de datos por pantalla. Además, tienen que cumplir ciertas condiciones entre ellas la de poder adaptarse en filas y columnas al número de elementos que en todos los casos dependen directamente de los grados de libertad.

Para completar las tablas al igual que con las gráficas también se ha usado la función 'ColumDataSource' [18] para que el código interprete la información. El problema en este caso surge cuando las filas o cadenas de caracteres que completan la tabla cambian en valor y tamaño, para esto, la forma en la que se rellena la información se ha desarrollado en una función que se puede llamar siempre los datos cambian y así actualizar el tamaño de la tabla y la información de ésta.

```
source_tab2 = ColumnDataSource(data=dict())
def update_source_tab2():
    global X0_des,X0_vel,X0
    data2 = dict( FF=[F[i] for i in range(n)],
                  ws1=[ws[i] for i in range(n)],
                  desf=[Desf[i] for i in range(n)],
                  X0_des=[X0[i] for i in range(0,n)],
                  X0_vel=[X0[i] for i in range(n,2*n)])
    source_tab2.data = data2
update_source_tab2()
```

Figura 4.9: Ejemplo de funcion que completa los datos de una tabla.

La función 'global' que se ve en la Figura 4.9, se encarga de llamar a un parámetro desde dentro de una función para que el código sepa que ese parámetro no es nuevo y único dentro de la función, sino que hace referencia a uno del código principal, esto es importante por si ese parámetro cambia, que dentro de la función tome el nuevo valor y viceversa.

En el tema de las columnas, por cómo está construido en Bokeh, estas son invariables para una tabla a diferencia de las filas que se pueden actualizar con la información como se veía arriba. Para las columnas se usa la función 'TableColumn' [18] donde se

crea cada columna de manera independiente y se le asigna un título o cabecera, se ve un ejemplo de esta función en la Figura 4.10.

```
columns2 = [TableColumn(field = "FF", title = "Amplitudes"),
            TableColumn(field = "ws1", title = "Frecuencias"),
            TableColumn(field = "desf", title = "Desfases"),
            TableColumn(field = "X0_des", title = "Desplazamiento Inicial"),
            TableColumn(field = "X0_vel", title = "Velocidad Inicial")]
```

Figura 4.10: Ejemplo de la función TableColumn.

Para los casos de los autovalores y autovectores, aparece un problema nuevo como evolución de cargar los datos en las tablas con la función 'ColumDataSource' [18], el problema con estos no es solo que se actualicen los datos de las tablas o la longitud de las cadenas de valores (número de filas), sino que no siempre van ser las mismas columnas, por ello es necesario desarrollar un bucle que reescriba la función con la finalidad de que estas cambien según sea el tamaño de las columnas de la tabla. El bucle que se ve en la Figura 4.12 permite reescribir la función creando los nombres de las columnas para que la misma función pueda cargar los datos en esta y que la tabla se cree completa.

```
source_tab4 = ColumnDataSource(data=dict())
def update_source_tab4():
    data4 = dict()
    for i in range(0, n):
        data4['vr'+str(i+1)] = [vr[a][i] for a in range(n)]
    source_tab4.data = data4
update_source_tab4()
```

Figura 4.11: Solución para reescribir una funcion automaticamente.

Ahora la función importante en este apartado, comparándola con la función 'figure' en el apartado de gráficas, es 'DataTable' [18]. Esta función es la encargada de crear las tablas, también controla los parámetros sobre tamaño y opciones como la de si se desea que el usuario pueda editar la tabla.

Uno de los problemas encontrados durante el desarrollo del código ha sido la excesiva exactitud de los resultados en los autovalores y autovectores pues Python trabaja con todo el numero completo y el número de decimales al que se tienen que adaptar las tablas es enorme, por esto en estas tablas de salida de datos se implementan bucles, como el de la Figura 4.12, que devuelvan los numero redondeados con los números necesarios.

```
for i in range(0,n):
    w1[i] = round(w[i], 3)
    sr1[i] = round(sr1[i], 3)
    dr1[i] = round(dr1[i], 3)
return sr1,dr1,w1
```

Figura 4.12: Solución del problema con los decimales.

Para el caso de la salida de datos de los autovalores y autovectores por pantalla, las tablas salen todas juntas y no tienen la opción de añadirlas un título por lo que para diferenciarlas entre ellas se usa la función 'PreText' [18] que deja añadir un título a mayores en el "layout" del programa y al principio de cada tabla.

4.6 Funciones de control (callbacks)

Frente a los cambios en los parámetros o propiedades a través de la interfaz el código necesita saber cómo reaccionar, para ello Bokeh dispone de funciones, que se pueden definir como 'Callbacks' [18], que permiten personalizar los comportamientos del código frente a esos cambios. Para cada cambio es necesario crear un patrón de comportamiento que se engloba en una función independiente que queda inactiva hasta que se la necesita y se ejecuta.

4.6.1 Descarga de datos

Una herramienta que necesitaba el programa desde el inicio de su desarrollo fue la de tener la opción de descargar los datos del mismo en un archivo compatible con el resto de programas. En este trabajo en concreto era necesario el poder comparar los resultados entre los de un ensayo real y los obtenidos en el programa con los datos estimados. Este "callback" [18] difiere del resto por tener una estructura diferente.

Para activar esta opción de descarga se hace desde un botón en la parte inferior de la columna de la izquierda. Este botón llama a una función auxiliar que está fuera del código llamada 'download.js', esta se ha obtenido de la página oficial de Bokeh y se ha modificado ligeramente para adaptarla a este proyecto. Las ventajas de trabajar con una librería de código abierto es poder acceder a funciones que ya están creadas, probadas y son gratuitas, sin necesidad de escribirlas desde cero ya que no están relacionadas con tema de este trabajo.

Para la programación de este apartado ha sido necesario implementar JavaScript pues, aunque Bokeh está preparado para el uso en web, tiene sus limitaciones y para ello da opción de recurrir a este otro lenguaje más específico para ello, se puede ver

en la Figura 4.13 el uso del mismo en las últimas líneas de código. Es necesario llamar al bloque 'CustomJS' [18] de entre los módulos de Bokeh.

```
descarga_1 = ColumnDataSource(data=dict())
def descarga():
    global descarga_1
    data_desc= dict(Vector_tiempo = [round(T_out [i], 6)  for i in range(T_out.shape[0])],
                    Respuesta_temporal = [round(y1 [i], 8)  for i in range(T_out.shape[0])],
                    Vector_frecuencia = [round(wt [i], 8)  for i in range(Num_elem)],
                    Magnitud_respuesta_frecuencia = [round(H1 [i], 8)  for i in range(Num_elem)],
                    Fase_respuesta_frecuencia = [round(F1 [i], 8)  for i in range(Num_elem)])
    descarga_1.data = data_desc
descarga()
btn.js_on_click(CustomJS(args=dict(source=descarga_1),
                         code=open(join(dirname(__file__),
                                "download.js")).read()))
```

Figura 4.13: Callback de descarga de datos.

El código está programado para devolver la información en formato "CSV" siendo este un sistema muy sencillo de almacenamiento de datos que gracias a esto puede ser interpretado por muchos programas. Como problema, a la hora de interpretar estos datos con programas como Excel el código descarga el valor completo de los números con todos sus decimales y separados por punto, pues es la forma de trabajar de Python, pero Excel no interpreta bien este formato y descoloca el resultado. Para solucionar el problema se limita el número de decimales que el código devuelve en la descarga. Esto se puede ver en la Figura 4.13.

También se ha implementado en una función la parte que rellena las listas de datos de la información que descargar, esto ayuda a poder llamar a la función de nuevo siempre que se actualicen los datos de salida con el simple hecho de llamar a la función con el nombre, 'descarga ()', como se ve en la Figura 4.13 después de la misma.

4.6.2 Cambio en la salida de la respuesta temporal

Este "callback" se ejecuta una vez que la herramienta de tipo "Select" [18] llamada "tipo", la que se muestra en la Figura 4.4, sufre un cambio siendo elegida otra opción entre las que aparecen en el listado de opciones. El "callback", una vez se ejecuta, se encarga de analizar cuál de las opciones es la que está activa, entonces completa los vectores encargados de definir la salida de la respuesta temporal (Posic, Veloc y Acel) que se ven en el Apartado 3.3.2. Una vez actualizados esos vectores se llaman, desde dentro de la propia función, de nuevo a las funciones sobre el análisis modal encargadas de ser el motor de cálculo para volver a calcular los valores de salida, entonces se actualizan los datos de las gráficas y del archivo de descargas.

Un problema que aparece con el desarrollo de esta función y que afecta al resto, es en la selección de una de las columnas del vector que alberga los datos sobre desplazamiento, velocidad o aceleración, siendo cada columna la información sobre un nodo en concreto. El problema aparece cuando el número de grados de libertad es uno y por la forma en la que está programado Python no entiende que le pidas una columna de un listado de datos (Array) que solo tiene una columna. Por ello se soluciona implementando unas líneas de código que primero comprueban el tamaño que tendrá el listado y entonces le pide una columna de datos en concreto o el listado entero.

```
if gdl_vis > 0:
    y1=y[:,gdl_vis]
if gdl_vis == 0:
    y1=y
```

Figura 4.14:Solucion al problema de listados de datos de una sola columna.

4.6.3 Cambio en el rango del tiempo o de la frecuencia

Si el usuario decide cambiar el rango del vector del tiempo o de la frecuencia, se han implementado en el mismo bloque por compartir muchas líneas de código, es necesario recalcular todos los datos de las respuestas temporales y la respuesta en frecuencia con el fin de que la salida por pantalla se represente correctamente. Para ello el “callback” actualiza los valores de los vectores llamando solo a las funciones de análisis modal que se encargan de recalcular los datos de salida y por último actualiza los listados de datos de las gráficas.

4.6.4 Cambio en la salida de la respuesta en frecuencia

Al igual que en el apartado 4.6.2, este “callback” se encarga de recalcular los datos cuando el usuario busca una salida diferente a la que se muestra por pantalla en lo que refiere a la respuesta en frecuencia, en concreto si la salida es sobre receptancias, movilidad o acelerancias. Para que este “callback” se ejecute basta con que sufran un cambio las herramientas sobre el tipo de salida, el vector de frecuencia o el nodo que se desea visualizar en las salidas por pantalla.

4.6.5 Cambio del número de grados de libertad

Este “callback” es quizá el más importante, hace referencia al número de grados de libertad y se ejecuta cuando cambia ese número. El código completo está basado en el número de grados de libertad que influye en el tamaño de las tablas, los límites de herramientas como los “spinners” [18] y lo más importante, en el tamaño de casi todos los vectores de información que repercuten también en lo anterior y en las salidas por pantalla como las gráficas.

```
if new > old:
    a = new
    b = old
    e = int(a - b)
    for i in range(0, e):
        m = np.append(m,1)
        c = np.append(c,0.001)
        k = np.append(k,1)
        Posic = np.append(Posic,Posic[0])
        Veloc = np.append(Veloc,Veloc[0])
        Acel = np.append(Acel,Acel[0])
        F = np.append(F,1)# Amplitudes
        ws = np.append(ws,1)# Frecuencias
        Desf = np.append(Desf,1)# Desfases
        X0 = np.append(X0,0)# Condiciones iniciales
        X0 = np.append(X0,0)# Dos veces por ser de dimension 2*n
```

Figura 4.15:Codigo parcial del callback sobre el grado de libertad.

El programa completo está estructurado de tal forma que al iniciarse se asignan unos valores por defecto para que el programa esté funcionando desde el principio, con esta mecánica se desarrolla una función que, a estos valores o a los que el usuario haya modificado, se les completa con valores aleatorios para que cumplan con el tamaño necesario y así el programa pueda funcionar sin problemas. También, la función elimina elementos de los datos si el usuario incluye un grado de libertad menor al existente en ese momento.

En la Figura 4.15 se muestra parte del código del “callback” por si cambia el grado de libertad del sistema, en concreto la parte que corresponde al bucle que añade elementos por si el grado de libertad aumenta respecto al que hay en ese momento, usando la función “numpay.append” [19]. En la figura también se puede ver todos los parámetros que es necesario modificar. El bucle que se encarga de eliminar números para ajustar el tamaño de los parámetros es similar al de la Figura 4.15, solo que hace uso de la función “numpay.delete” [20] eliminando por defecto el primer número del vector.

Ajustado los parámetros para que los tamaños coincidan, el “callback” llama de nuevo a todas las funciones de la parte de cálculo modal, es necesario recalcular todos los

datos del programa. Es importante usar la función 'global' antes mencionada para que todos los parámetros queden actualizados fuera de esta función.

Una vez actualizado el tamaño de los parámetros y los valores de los mismos, es momento de actualizar todos los valores de las tablas, gráficas y parámetros de las herramientas como el valor por defecto o el rango de valores en los "spinners" [18]. Aquí se ve la ventaja de haber incluido en funciones independientes la manera en las que las tablas y graficas toman los datos, pues ahora basta con llamar de nuevo a esas funciones para que actualicen los datos.

4.6.6 Actualizar las entradas de datos por tabla

Si el usuario decide cambiar los valores de ciertos parámetros como las condiciones iniciales o los valores de las rigideces lo puede hacer modificándolos directamente sobre las tablas y, una vez que ha terminado de modificarles todos, pulsar sobre el botón de actualizar para que el programa recalcule. La función de este "callback" es la de coger los datos que han cambiado, cargarlos en los parámetros del programa y recalcular.

Aparece un problema como limitación de las funciones de Bokeh, pues no permite extraer un solo elemento localizado de las tablas, entonces la función, una vez se pulsa el botón de actualizar, descarga todos los valores que estén en ese momento en la tabla y los carga en los parámetros de nuevo, quedando así actualizados. Esto se ve en la Figura 4.16, que muestra parte del código donde se realiza esta tarea.

```
m = np.array(source_tab1.data['mm'], dtype=np.float)
c = np.array(source_tab1.data['cc'], dtype=np.float)
k = np.array(source_tab1.data['kk'], dtype=np.float)
F = np.array(source_tab2.data["FF"], dtype=np.float)
ws = np.array(source_tab2.data["ws1"], dtype=np.float)
Desf = np.array(source_tab2.data["desf"], dtype=np.float)
X0_des = np.array(source_tab2.data["X0_des"], dtype=np.float)
X0_vel = np.array(source_tab2.data["X0_vel"], dtype=np.float)
X0 = np.concatenate((X0_des, X0_vel), axis=0)
```

Figura 4.16: Codigo parcial del callback de los datos de la tabla.

Por último y como el resto de "callbacks", vuelve a recalcular los valores llamando a las funciones del cálculo modal y se actualizan los datos del resto de tablas y de las gráficas.

4.7 Ordenes de diseño para el layout

Finalmente, ya están creadas todas las herramientas, tablas y gráficas, además todas tienen funcionalidad y están bien relacionadas entre ellas. Por lo tanto, solo queda decirle al programa, a Bokeh, como debe distribuir esos elementos en el espacio de la pantalla. Como se veía en el apartado de disposición y layout, la estructura principal de la interfaz serán dos columnas.

La primera columna contiene todos los datos comunes a todo el programa sobre parámetros y entradas y salidas de información. Se hace una separación en esta columna usando la función 'Tabs' de Bokeh, esta permite crear una ventana que se puede cambiar entera mediante diferentes pestañas, con esto se consigue separar algunos parámetros que solo interfieren en la parte de respuesta temporal y otros que solo intervienen en la de respuesta en frecuencia. Dentro de esta ventana los elementos se han distribuido en dos filas, estas cambian según la pestaña.

La segunda columna, la de la derecha, entera se ha programado con la función 'Tabs' entera, esto es porque dependiendo de la salida que quiera el usuario sacar por pantalla, la columna cambia entera. Esta columna se encarga de mostrar los datos que salen por pantalla como respuesta del sistema.

```
tab1 = Panel(child=colum_tiempo, title="Tiempo")
tab2 = Panel(child=colum_freq, title="Frecuencia")
tab3 = Panel(child=colum_modal_autovalor, title="Cálculo modal")
tabs = Tabs(tabs=[ tab1, tab2 , tab3])
```

Figura 4.17: Ejemplo de la funcion Tabs de Bokeh.

Capítulo V. Ensayo Experimental

5.1 Introducción

La finalidad de este capítulo es ensayar una maqueta y obtener los resultados sobre su comportamiento frente a unas condiciones iniciales, para más adelante, estimar y obtener los mismos resultados mediante el programa desarrollado con el fin de validar el código. Una vez se ha terminado el desarrollo del código y de la interfaz hay parámetros que son necesarios ajustar y comprobar, siendo la mejor manera con un modelo real sobre el que se conoce todos esos datos. Además de demostrar su funcionalidad y utilidad en un caso real.

5.2 Ensayo de laboratorio

La simulación de la maqueta se ha realizado en el Laboratorio de Estructuras en la sede de Paseo del Cauce de la Universidad de Valladolid. La maqueta consiste en una estructura de dos grados de libertad compuesta por láminas de aluminio como pilares y placas de metacrilato como base de las plantas, como se ve en la Figuras 5.1.

Figuras 5.1: Maqueta del ensayo experimental

El código del programa permite hacer diferentes tipos de ensayos, entre ellos la aplicación de una fuerza senoidal conocida o la aplicación de unas condiciones iniciales y el estudio del comportamiento del sistema hasta su posicion de reposo. La primera opcion es más complicada de aplicar, por la dificultad de reproducir las fuerzas. La segunda es más viable, facil de repetir y disponer de varias muestras de datos. Una vez se ha decidido por la segunda, se dispone la maqueta con distintos sensores en los nodos que interesa estudiar.

La forma de proceder es provocar a la estructura unas condiciones iniciales que son conocidas y medir los desplazamientos de las diferentes plantas de la estructura, siendo estas los nodos. Para proceder con el ensayo es suficiente con mover manualmente la parte superior de la estructura, despues de haber anclado la misma al suelo y mantenerla lo mas inmovil posible en lo que la toma de datos empieza. La posicion inicial no es relevante encuanto que se contrastara la misma en el programa basado en el codigo mediante el vibrómetro laser. Es importante destacar que la velocidad inicial es nula, esto es porque no se aplica ninguna furza a la estructura, simplemente se posiciona en las condiciones iniciales sin más. Se repite el ensayo con el fin de tener varias muestras.

5.2.1 Equipo usado en el ensayo

Para la toma datos es necesario disponer de varios sensores, esto es un problema porque pueden no estar sincronizados. Este problema puede provocar la superposición de datos o que al contrastar el tiempo en el que se ha tomado cada dato, estos no coinciden. Para ello el elemento más importante en el experimento es el sistema Sirius DAQ.

- **Sirius DAQ**

  Este instrumento es un sistema de adquisición de datos que permite unir todos los sensores a usar en las diferentes entradas, independientemente de la diferencia de sensores y voltajes proporcionados por cada sensor para la señal, uniéndolo todo en una sola salida que es interpretada por el propio software del Sirius.

Figuras 5.2: Sistema de adquisición de datos Sirius

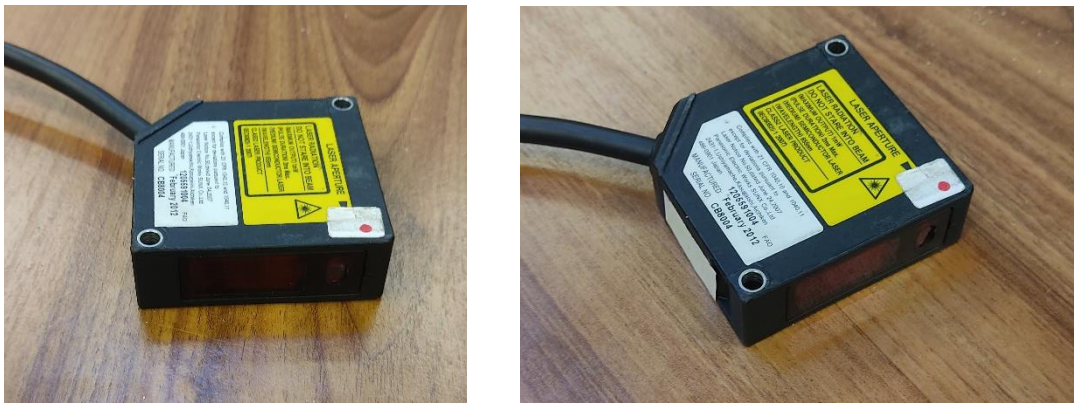
Los diferentes conectores que se aprecian en la primera imagen de las Figuras 5.2 son los puertos de entrada de los que dispone el Sirius, siendo necesario uno por sensor. No todos los sensores disponen de este tipo de puerto por lo que se usa un adaptador de VGA en los que son necesarios, como se ve en la Figura 5.3.

Figuras 5.3: Conexiones al Sirius

Listo el sistema de adquisición de datos es necesario definir qué tipo de sensores son los más adecuados. Es importante medir los desplazamientos de los nodos, por ello se usa un vibrómetro.

- **Vibrómetro Laser (Desplazamiento)**

La ventaja de un vibrómetro frente a otro tipo de sensor es que este permite medir a una alta frecuencia de muestreo y proporcionara un mejor resultado. El usado en el ensayo es el que se muestra en la figura 5.4. Para la disposición de este sensor, la referencia de las medidas depende de la posición espacial a la que se instale, contando con la distancia que le separa a la estructura desde un inicio. Es importante tenerlo en cuenta a la hora de analizar las gráficas pues la posición de reposo no será la del cero como puede ser la del programa basado en el código.



Figuras 5.4: Vibrómetro Laser

Para la colocación de este sensor se usa una estructura auxiliar para sostenerlo de la forma más inmóvil posible, pues funciona por desplazamientos, colocándolo a la altura del nodo superior. Esto se puede apreciar en la Figura 5.1. Solo se usa un sensor para el desplazamiento del segundo nodo pues del primero solo se necesita el desplazamiento inicial para las condiciones iniciales y este se puede estimar.

Como complemento al vibrómetro laser, se usan dos acelerómetros para los dos nodos del sistema y medir en el tiempo las aceleraciones durante el ensayo. Estas también se pueden contrastar con la información del programa.

- Acelerómetro piezoeléctrico (Aceleración)

Estos acelerómetros miden la aceleración del sistema al que estén acoplados mediante señales eléctricas que son interpretas por el sistema de adquisición de datos y su programa mediante una calibración previa que suele proporcionar el fabricante o mediante el uso de otros sensores.

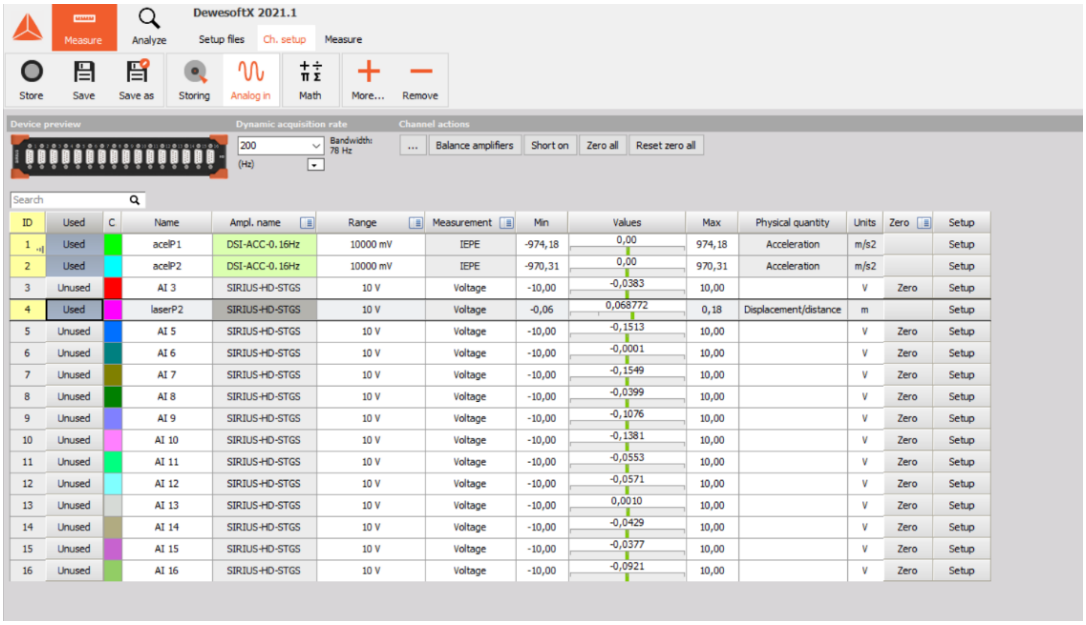
Figura 5.5: Acelerómetro piezoeléctrico

Para el vibrómetro laser bastaba con sujetar el sensor en la estructura fija, pero en el caso del acelerómetro es necesario que esté acoplado a la estructura, más concretamente en los nodos a estudiar. Para la sujeción del sensor se usa un imán al que por un lado se acopla el sensor y por el otro se une a la estructura, como se ve en la Figura 5.6, esta sujeción es más que suficiente para realizar el ensayo. El imán no afecta al funcionamiento del acelerómetro pues este tipo de sensores funcionan por la compresión de un retículo cristalino piezoeléctrico que genera una señal eléctrica proporcional a la fuerza sufrida por el sensor.

Figura 5.6: Acople del Acelerómetro a la estructura

5.2.2 Calibración del programa del Sirius

Una vez está lista la parte más manual del ensayo es momento de realizar las mediciones, pero antes, se necesita preparar el programa y las respectivas calibraciones. El programa usado para interpretar las señales del Sirius es el proporcionado por el fabricante, siendo el DewesoftX 2021. Lista la correcta instalación de los cables entre los sensores, el Sirius y el ordenador es momento de comprobar que el programa detecta todos los sensores. Como se ve en la Figura 5.7.



ID	Used	C	Name	Ampl. name	Range	Measurement	Min	Values	Max	Physical quantity	Units	Zero	Setup
1	Used		acelP1	DSI-ACC-0.16Hz	10000 mV	IEPE	-974,18	0,00	974,18	Acceleration	m/s2		Setup
2	Used		acelP2	DSI-ACC-0.16Hz	10000 mV	IEPE	-970,31	0,00	970,31	Acceleration	m/s2		Setup
3	Unused		AI 3	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0383	10,00		V	Zero	Setup
4	Used		laserP2	SIRIUS-HD-STGS	10 V	Voltage	-0,06	0,068772	0,18	Displacement/distance	m		Setup
5	Unused		AI 5	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,1513	10,00		V	Zero	Setup
6	Unused		AI 6	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0001	10,00		V	Zero	Setup
7	Unused		AI 7	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,1549	10,00		V	Zero	Setup
8	Unused		AI 8	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0399	10,00		V	Zero	Setup
9	Unused		AI 9	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,1076	10,00		V	Zero	Setup
10	Unused		AI 10	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,1381	10,00		V	Zero	Setup
11	Unused		AI 11	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0553	10,00		V	Zero	Setup
12	Unused		AI 12	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0571	10,00		V	Zero	Setup
13	Unused		AI 13	SIRIUS-HD-STGS	10 V	Voltage	-10,00	0,0010	10,00		V	Zero	Setup
14	Unused		AI 14	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0429	10,00		V	Zero	Setup
15	Unused		AI 15	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0377	10,00		V	Zero	Setup
16	Unused		AI 16	SIRIUS-HD-STGS	10 V	Voltage	-10,00	-0,0921	10,00		V	Zero	Setup

Figura 5.7: Configuración canales del sirius

Como se ve en la figura 5.7, directamente al entrar en el programa se ven los diferentes puertos del Sirius, entre ellos a los que están conectados los sensores del ensayo. El programa detecta automáticamente ciertas características de los sensores

como el rango de valores o el nombre, aunque algunas tienen que ser ajustadas a mano.

Un parámetro que es importante tener en cuenta ahora y en el programa basado en el código es la frecuencia de muestreo. Esta es importante que sea los más parecida entre las dos fuentes de información para que la comparación sea más exacta. En las opciones principales del programa del Sirius, parte superior de la Figura 5.7, se puede modificar siendo en este ensayo de doscientos la frecuencia de muestreo.

Los sensores no proporcionan la información de una forma directa, lo que envían son señales eléctricas que deben ser interpretadas, siendo para cada sensor diferente al resto. Para que el programa interprete correctamente estas señales es necesario calibrarlo indicando las equivalencias de las mismas a datos de interés, accediendo desde el botón de configuración de cada canal como se ven en la Figura 5.7. En la Figura 5.8, en la sección inferior recuadrada de verde, se ve la calibración necesaria para el primer acelerómetro dispuesto en el primer canal del Sirius como ejemplo. Los datos para calibrar cada sensor suelen ser proporcionados por el fabricante, también se pueden calibrar con otro sensor ya calibrado.

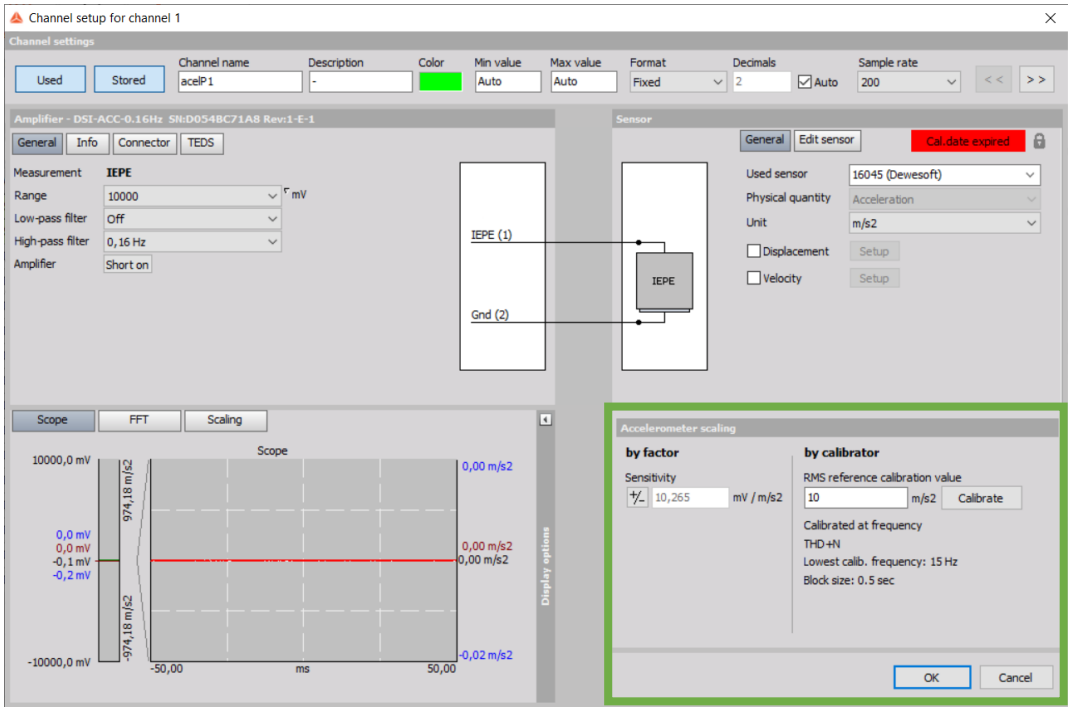


Figura 5.8: Calibración de un acelerometro.

Para el sensor de los desplazamientos hay una excepción, este sensor mide la distancia directamente desde su posición a la posición del nodo sobre el que mide por ser él mismo su propia referencia. El problema aparece cuando se contrasta la información con otras fuentes, pues en el caso del programa del código desarrollado mide directamente los desplazamientos desde la posición de reposo y al comparar ambas aparece un desfase en el ensayo experimental proporcional a la distancia a la

que se ha colocado el sensor laser de la estructura a ensayar. La solución a este problema la da directamente el programa DewesoftX, este permite la creación de un canal virtual al que se le puede configurar de tal forma que a las mediciones del sensor laser de los desplazamientos les reste o sume el desfase. En este caso se ve como debajo del título de 'Value' en la Figura 5.9 resta el valor del desfase a la medida 'laserP2' con el fin de que la posición de reposo de las medidas coincida con el de referencia buscado.

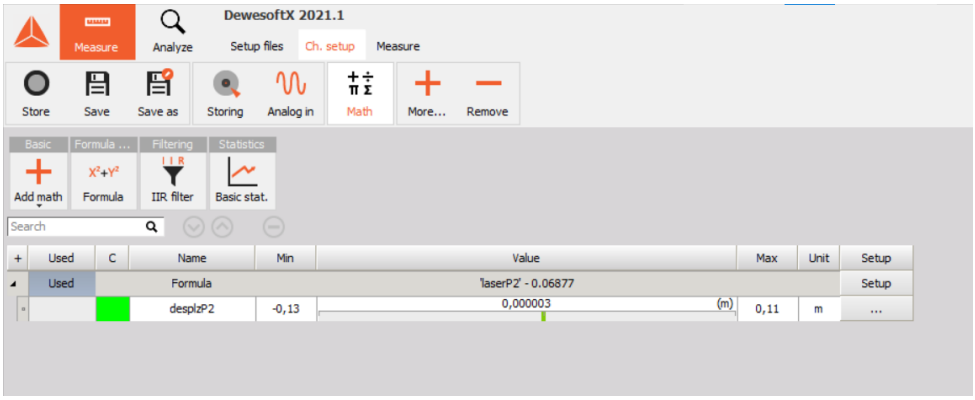


Figura 5.9: Canal virtual en DewesoftX.

El desfase es fácil de sacar para este caso, basta con graficar los datos y estudiar la posición de reposo, al ser cero la que se busca, el desfase será la diferencia.

5.2.3 Recogida de datos

Finalizada la calibración de sensores en el programa del Sirius, empieza la toma de datos. Las condiciones iniciales serán cero para las velocidades y el valor máximo que registren los sensores para los desplazamientos, pues la forma de realizar el ensayo es separando al nodo superior de su posición de equilibrio y soltar de repente, simulando las condiciones iniciales fuera de las de reposo. Una vez el ensayo empieza a ser grabado, dura hasta que la estructura llega a su posición de reposo y se repite con el fin de tener varias muestras para estudiar.

En la figura 2.10, se puede ver las diferentes salidas que se le pide al programa que saque por pantalla, en el que la gráfica verde es la salida del acelerómetro colocado en la primera planta, la azul la de la segunda planta y la roja es la salida del sensor laser que mide el desplazamiento de la segunda planta. La salida directa del vibrómetro no es necesaria pues es la misma del canal virtual con la ventaja de que la del canal virtual está ya corregida. Se aprecia como los acelerómetros empiezan ambos en cero y el sensor de desplazamientos empieza en la posición de máximo desplazamiento por las condiciones iniciales.

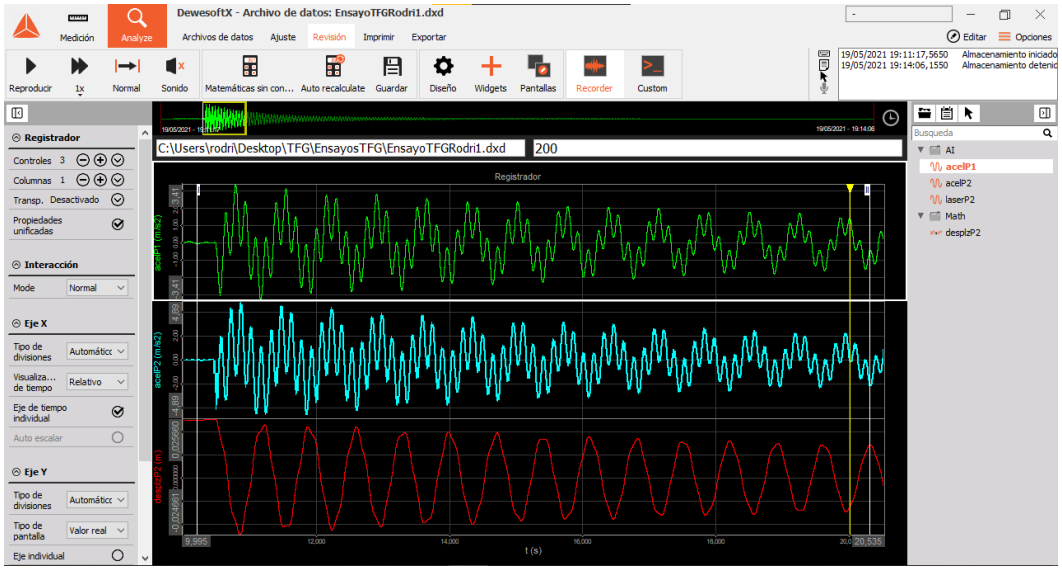


Figura 5.10: Salida de datos en el Ensayo.

Desde el mismo programa se puede descargar los datos en un archivo de Excel, siendo la forma fácil de poder contrastarlos con los de la estimación ya que también puede descargar los datos en un archivo compatible.

Cabe destacar que los valores positivos y negativos de las gráficas hacen referencia a los ejes de los sensores que dependen de la orientación de los mismos. Siendo los ejes los que se muestran en la Figura 5.11, los elementos rojos son los dos acelerómetros y el elemento verde el sensor laser para los desplazamientos, cada uno orientado de forma que los ejes horizontales coincidan en el mismo sentido.

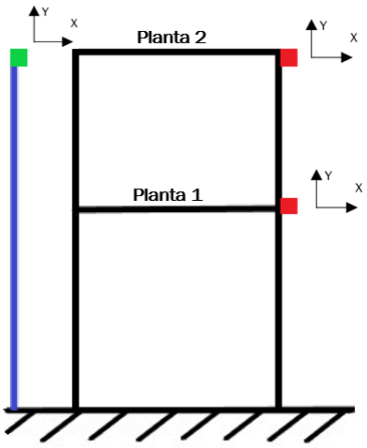


Figura 5.11: Esquema de la estructura ensayada.

El programa también dispone de la opción de estimar las frecuencias propias de la estructura, estos valores son una referencia para comparar los resultados del ensayo con los de la estimación del programa basado en el código.

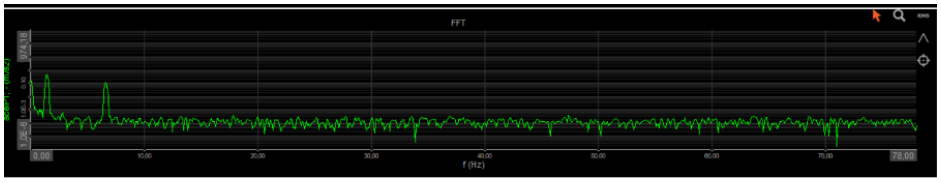


Figura 5.12: Estimación de las frecuencias propias.

De la Figura 5.12 salen los valores de las dos frecuencias propias, siendo dos por ser un sistema de dos grados de libertad. Los valores estimados son los picos de máxima amplitud, siendo 1,465 Hz y 6,641 Hz respectivamente.

5.3 Estimación de valores

Para sacar resultados del programa, se necesitan todos los datos sobre las matrices de masa y rigidez pudiendo hacer aproximaciones por las dimensiones y características de la estructura. Además, también hacen falta los datos sobre las condiciones iniciales de los dos nodos, teniendo que estimar la del primer nodo por no tener un sensor en este. Para empezar, se aproxima el valor de las masas de los dos nodos que componen la estructura y los valores sobre las rigideces que corresponden a los pilares, luego sus correspondientes rigideces y por último el valor de las condiciones iniciales.

5.3.1 Estimación de las masas

La estructura está compuesta de dos materiales diferentes, siendo aluminio las dos placas laterales que hacen de pilares en la estructura y metacrilato el de las entreplantas primera y segunda. Las densidades de estos materiales son de 2700 kg/m^3 y 1200 kg/m^3 respectivamente. Junto a las dimensiones de la estructura se calcula el peso correspondiente al primer nodo, al que le corresponden las dos planchas de aluminio hasta la primera planta y el bloque de metacrilato de la misma. El resto de la estructura corresponde a la masa del segundo nodo. Las dimensiones de la estructurase pueden ver en la Figura 5.13.

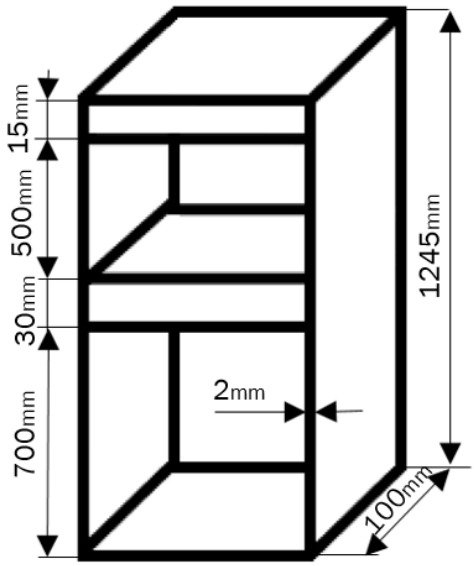


Figura 5.13: Dimensiones de la estructura ensayada.

Una vez se resuelven las ecuaciones, las masas estimadas son de 1,8 Kg la primera y 1,1 Kg. A esta estimación no se le ha sumado los mecanismos de anclaje de la estructura misma ni el peso de los sensores que lleva acoplados por ser datos desconocidos. Esto es necesario tenerlo en cuenta para posibles desviaciones en los datos, aunque el peso total despreciado es pequeño en comparación con el del resto de la estructura.

5.3.2 Estimación las rigideces

Por la forma en la que trabajan los pilares, o placas de aluminio en este caso, la componente de la matriz de rigidez para barras rectas biempotradas es la que hace referencia al segundo término de la diagonal de la matriz de rigidez. Siendo la Ecuación 3.1.

$$K = \frac{12 \cdot E \cdot I}{L^3} \tag{3.1}$$

El valor del módulo de Young se conoce al conocer el material (E= $6{,}5 \cdot 10^{10}$ Pa), el momento de inercia se saca directamente con las dimensiones de la sección del perfil de la Figura 5.13 y la ecuación de momentos de inercia para secciones cuadradas (I=$6{,}6 \cdot 10^{-11}$ m^4) y por último la longitud del perfil depende del tramo que se calcule

El valor de la rigidez total para el primer nodo, teniendo en cuenta que son dos pletinas iguales y que sus rigideces se suman al estar en paralelo, se estima que es de 300 N/m y 1120 N/m para el segundo nodo.

5.3.3 Estimación de las condiciones iniciales

Como condiciones iniciales el programa necesita los desplazamientos de los dos nodos y sus velocidades, estas dos últimas son nulas por cómo se ha realizado el ensayo. El desplazamiento inicial de la segunda planta se conoce pues es el valor máximo de la amplitud en la gráfica que representa el desplazamiento de esta planta en el tiempo. Para el segundo se resuelve el sistema de la Ecuación 3.2.

$$\begin{bmatrix} F1 \\ F2 \end{bmatrix} = \begin{bmatrix} K_1 + K_2 & -K_2 \\ -K_2 & K_2 \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{3.2}$$

Basta con resolver la ecuación de la parte superior ya que los valores son conocidos siendo la fuerza en este nodo cero por no aplicarse ninguna, las rigideces son conocidas y el desplazamiento es el valor a calcular, estimando su valor en 0,021 m.

5.4 Comparación de resultados

La finalidad de este punto es conseguir los mismos resultados en el programa basado en el código desarrollado que en el ensayo experimental del laboratorio a base de modificar los valores estimados. Para hacer la primera estimación se introducen todos los datos estimados, en unidades internacionales, en el programa y se ajustan los parámetros necesarios para que las dos frecuencias naturales coincidan entre sí.

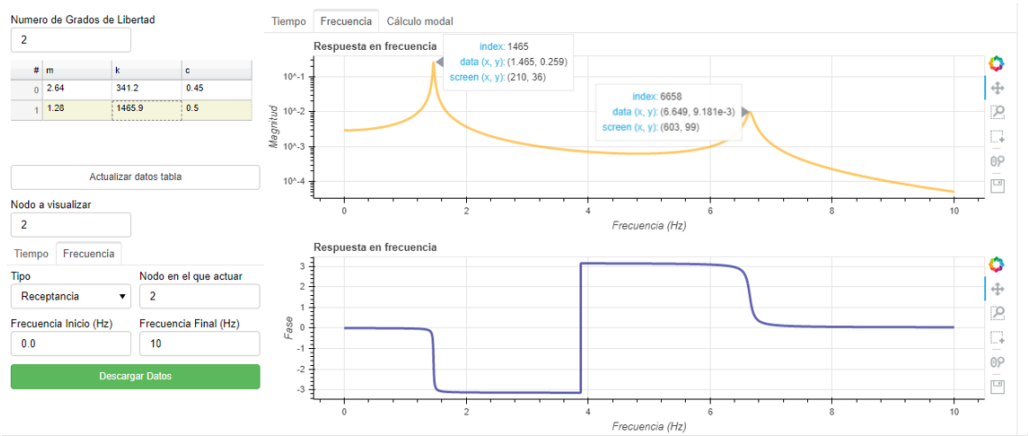


Figura 5.14: Frecuencias naturales obtenidas en el programa.

En la primera iteración, el error es de décimas, por lo que es necesario ajustar los valores introducidos para ajustar los valores. Tras modificar y ajustar los valores de las masas, rigideces y amortiguamiento, parámetros que se ven en la Figura 5.14, se llega a unos valores prácticamente iguales a los del ensayo experimental siendo 1,465 Hz y 6,649 Hz. Ahora, es necesario comprobar que el resto de salidas coincidan también. Los nuevos valores para las masas son 2,64 Kg para el primer grado de libertad y 1,28 Kg para el segundo, en cuanto a la rigidez es 341,2 N/m y 1465,9 N/m y

finalmente para el amortiguamiento es 0,45 N·s/m para el primer grado de libertad y 0,5 N·s/m para el segundo. Los valores difieren de las estimaciones iniciales, en el caso de las masas puede deberse a haber despreciado algunos elementos como tornillos, cables y aparatos de medida pues el programa usa valores mayores, en el caso de la rigidez puede deberse a factores como la temperatura o el estado de los materiales usados ya que para el experimento se han usado valores de las propiedades de los materiales normales a condiciones de temperatura ambiente.

Para comprobar como de cerca están los valores del ensayo real, el programa desarrollado permite descargar los datos en un formato compatible con Excel y así poder comparar los resultados entre ellos.

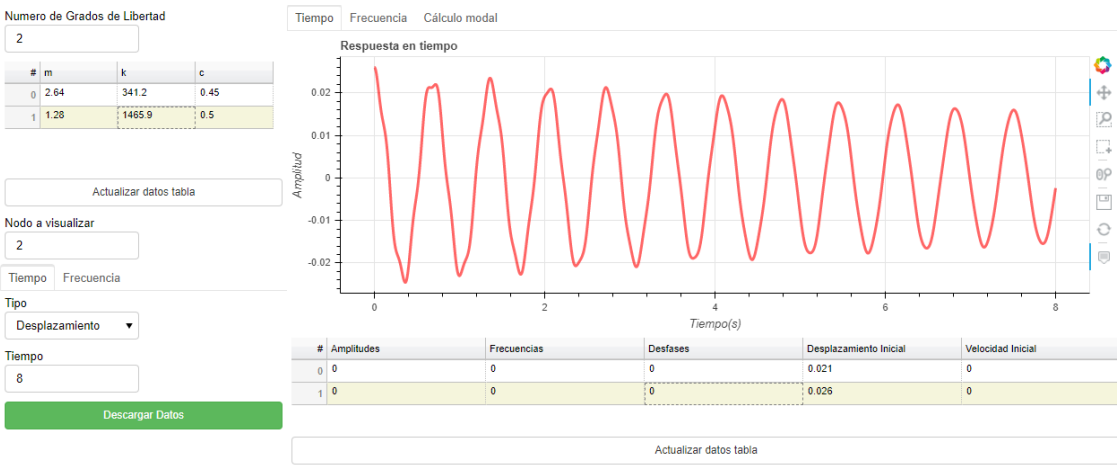


Figura 5.15: Prueba de salida de datos del programa.

Descargado los datos sobre el desplazamiento en el tiempo tanto del ensayo real como del programa, Figura 5.14, se pueden superponen como se ve en la Figura 5.15, siendo la gráfica azul el ensayo real y la roja los datos del programa.

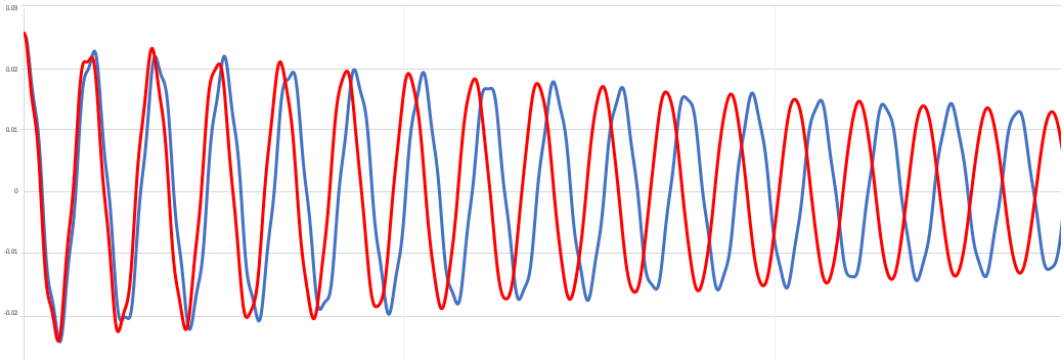


Figura 5.16: Comparación de resultados.

Aunque las primeras curvas parecen cuadran al principio, llevan un desfase en la frecuencia. Es necesario modificar en general los parámetros hasta conseguir que las curvas lleguen a coincidir entre ellas, al menos en frecuencia. Tras muchas pruebas y errores, se llega al resultado que se muestra en la Figura 5.17, apreciando que las curvas coinciden prácticamente en todo excepto ciertos matices.

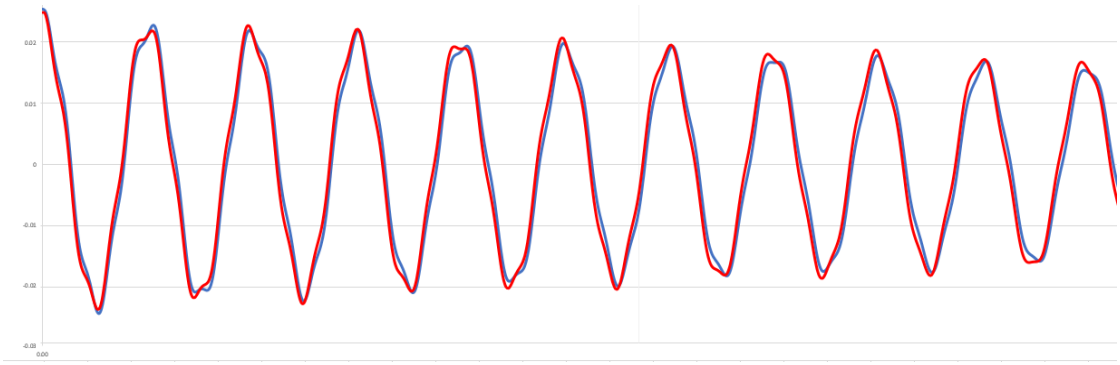


Figura 5.17: Comparación final de resultados.

Para llegar al resultado donde cuadran las gráficas de los desplazamientos del programa y el ensayo ha sido necesario sacrificar la exactitud de las frecuencias propias en unas décimas quedando en este caso 1,447 Hz y 6,638 Hz. Además, se ha incluido un valor, muy pequeño, para las velocidades pues en el caso del ensayo real este valor no llega a ser nulo por la forma en la que se ha procedido. Los parámetros finales son los que se ven en la Figura 5.18.

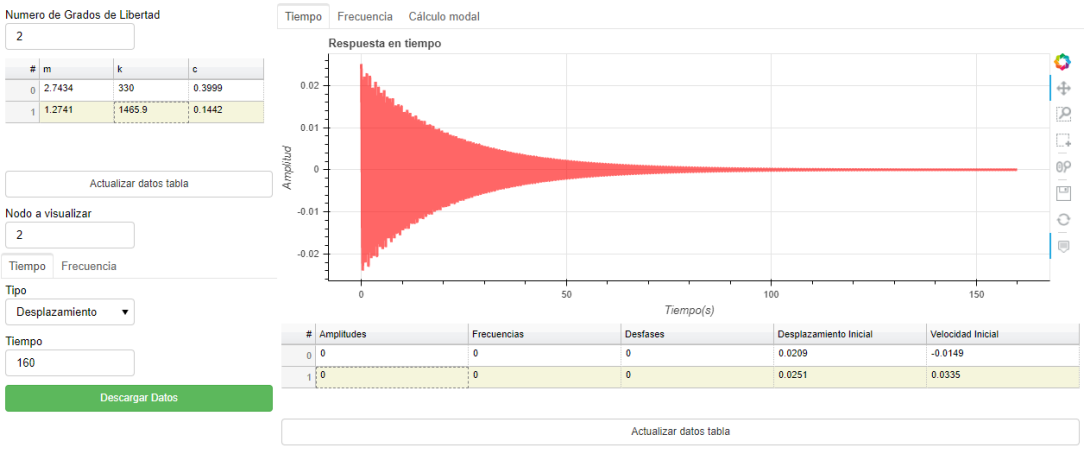


Figura 5.18: Parámetros finales de la comparacion.

Los resultados finales no son fieles al ensayo real como se puede ver en la Figura 5.19 donde se superponen las curvas completas de los desplazamientos frente al tiempo y se aprecia como hacia el final la curvatura de los datos del programa es menor que la del ensayo real, esto se debe al amortiguamiento.

El amortiguamiento estimado en el código se ha desarrollado de forma que sea lineal, el problema es en el caso real, pues aquí el amortiguamiento depende de más

condiciones, entre ellas el movimiento que lleva la estructura en cada momento. Programar estos cambios para que el código se adapte a los resultados es más complejo de lo que se estima para el programa, aportando poca información adicional.

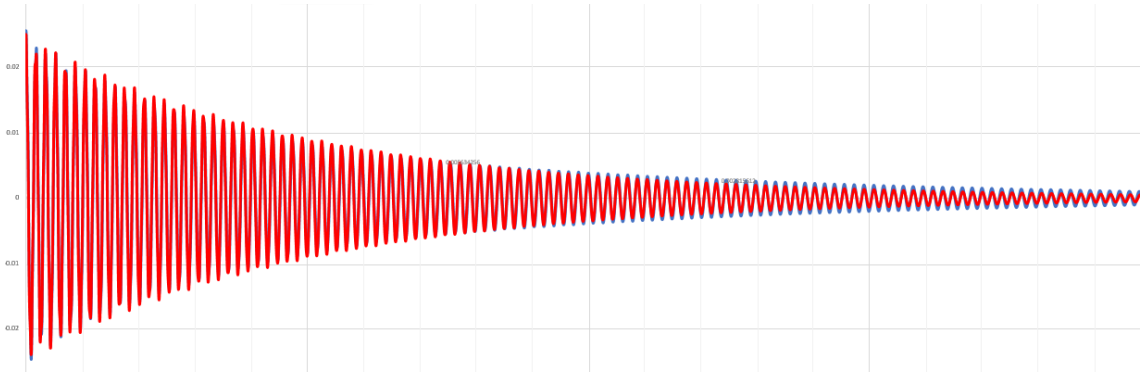


Figura 5.19: Graficas completas del desplazamiento en el tiempo.

Con esto se demuestra como el programa desarrollado permite estimar parámetros del ensayo con los análisis de un ensayo real. Esto es útil en los casos en los que se tienen los datos obtenidos de los sensores y se desconocen parámetros como el amortiguamiento, la rigidez, la masa o las condiciones iniciales.

Capítulo VI. Manual del Usuario

6.1 Introducción

Como en todo programa, es necesario un manual de uso. En este caso, esta parte es una propuesta de lo que puede incluirse en el apartado o pestaña de ayuda tanto como guía para el usuario como también un apartado donde resolver dudas de ciertos aspectos como la función de algunas opciones o la localización de funciones y ayuda en problemas comunes. En general este apartado se centrará en una descripción de la disposición del layout junto con una explicación de los diferentes menús, tareas comunes en el programa y funciones avanzadas que destacan el programa.

6.2 Opciones de acceso al programa

Este apartado corresponde a la parte sobre la guía de instalación que podría existir en un manual común, a diferencia de que una de las ventajas de este programa es la ausencia de instalación y ocupación de espacio en el ordenador del usuario pues se ha desarrollado de forma que el acceso sea remoto.

Como se ha dicho, este capítulo es una propuesta del manual de uso, por lo que este apartado puede variar una vez finalizado el presente trabajo y se prepare el programa para funcionar. La idea principal es que el usuario pueda acceder desde un ordenador o dispositivo similar a través de internet y use el programa con cualquier navegador. El programa funcionará en un ordenador desde la universidad a modo de servidor, esto es una ventaja frente a otros programas pues no necesita que el dispositivo desde el que se conecte el usuario cumpla unos requisitos mínimos, además de que no requiere una instalación previa ni archivos para la instalación. También destacar que, al ser el acceso desde un navegador de internet, en principio el sistema operativo del que disponga el usuario no es relevante, haciéndolo así más libre y abierto.

Se ha pensado que la forma de acceder sea mediante un enlace que se pueda encontrar en la página de la Universidad de Valladolid y además público para toda el que quiera hacer uso de él.

6.3 Aplicaciones del programa

El programa se ha desarrollado para el ámbito del análisis modal de sistemas discretos unidireccionales. La principal aplicación del programa consiste en comprender, describir y modelar el comportamiento del sistema frente a fuerzas externas que simulan casos de su vida útil en funcionamiento. Por ello el programa permite, entre otras funciones, la identificación de las frecuencias propias y modos de vibración del sistema para un rango de frecuencias mediante la información de las propiedades dinámicas de la estructura, esta información es importante durante el desarrollo y diseño del sistema pues ayuda a analizar cómo las fuerzas afectan a la estructura. También, al revés que antes, permite sacar datos sobre las propiedades dinámicas de una forma indirecta con datos como las frecuencias propias, así es como se ha procedido en el Capítulo 5 sobre el ensayo experimental.

El programa dispone de gráficas tanto de las funciones de respuesta en frecuencia como de la respuesta temporal. Estas son gráficas interactivas, disponen de opciones como visualizar los datos de ciertos puntos directamente sobre la gráfica, hacer zoom y moverla o incluso aislar parte de la gráfica para que solo enseñe esa porción que el usuario escoja. Además, todas esas gráficas son descargables de forma independiente

El usuario también tiene a disposición todos los datos que el programa ha procesado y con los que ha realizado las gráficas, estos al igual que las gráficas también son descargables. El programa descarga los datos en un archivo con extensión “CSV”, estas significan “Comma Separated Values” lo que quiere decir que entrega un listado de caracteres separados por comas, esto hace al archivo muy genérico y puede ser abierto desde Excel hasta los lectores de documentos más sencillos y comunes como los que contienen la extensión TXT en Windows.

El programa también dispone de algunas funciones especiales, los valores de los autovalores, autovectores y factores de amortiguamiento crítico. Entre las ventanas del programa está la de Cálculo Modal, aquí se encuentran a disposición del usuario tablas con todos estos valores tanto del caso amortiguado como del no amortiguado. Son valores que no siempre toman tanta importancia como las gráficas, las frecuencias propias y modos de vibración, pero son importantes y pueden ser de utilidad dando así un valor añadido al programa.

6.4 Descripción de los menús

Es importante una descripción de los menús que sirva de referencia para el usuario y explique todas las funciones del programa, desde las más obvias hasta las que son menos intuitivas. En el siguiente apartado se explicarán los menús con capturas del mismo y anotaciones sobre estas junto con leyendas que recojan toda la información.

6.4.1 Entrada de datos

Como se ha visto en el apartado 4.2 sobre la interfaz gráfica, la primera parte del programa a la que se tiene acceso es la columna de la izquierda sobre la entrada de parámetros. Es la que se ve en la Figura 6.1.

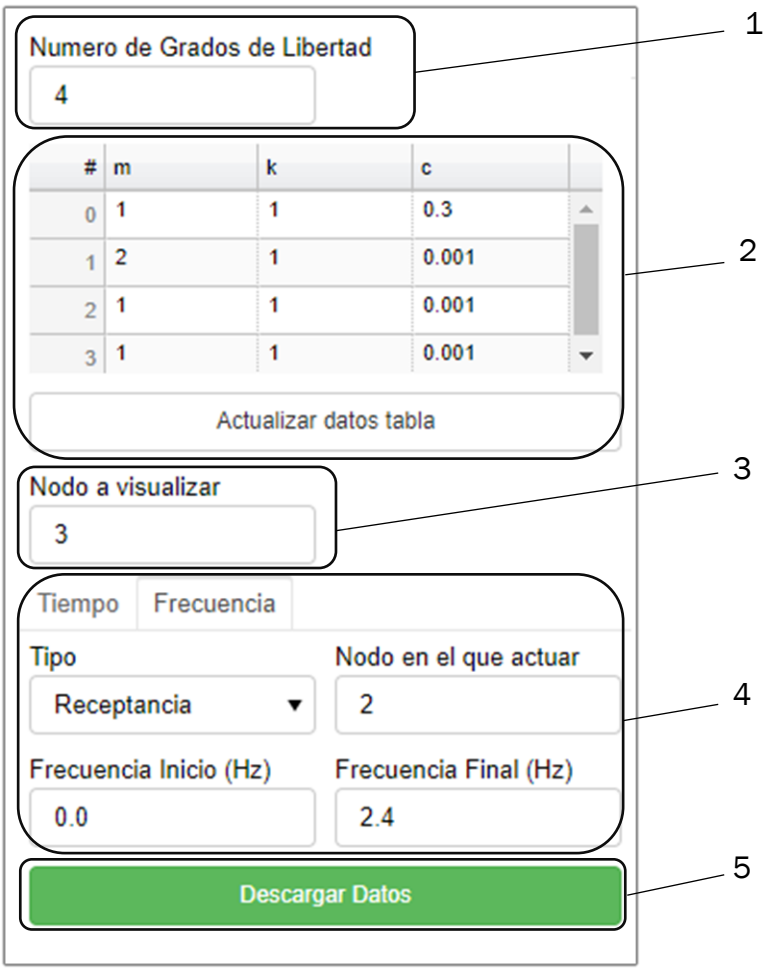


Figura 6.1: Entrada de parámetros en la interfaz gráfica

1. **Botón de grados de libertad:** Empezando por la primera ventana, esta muestra una de las opciones más importantes. De este botón dependen los demás parámetros del programa, incluso en tamaño y apariencia como es el caso de las tablas que se adaptan en tamaño para introducir todos los datos del sistema, por lo que es aconsejable que sea el primer dato en introducir.

2. **Tabla de propiedades dinámicas de la estructura:** La segunda ventana se centra en los datos que definen el sistema, divididos por columnas se introducen los datos de las matrices de masa, rigidez y amortiguamiento de los diferentes grados de libertad, divididos estos por filas. Una vez se ha realizado todos los cambios en la tabla, el botón de "Actualizar datos tabla" actualiza esos valores en el programa y recalcula de nuevo todos los resultados. El programa no lo hace de forma instantánea al introducir un nuevo valor para no tener que repetir el proceso varias veces si son muchos los datos nuevos.

3. **Grado de libertad a visualizar:** Esta pestaña muestra un contador o "spinner" [18] que selecciona el grado de libertad sobre el que se quieren mostrar los resultados y salidas por pantalla de las gráficas.

4. **Cuadro de control sobre respuesta temporal o en frecuencia:** Aquí aparece el primer apartado que varía totalmente según la pestaña que se seleccione. La primera pestaña, Tiempo, de las dos que se encuentran en la parte superior de esta ventana, da acceso a las opciones sobre la gráfica de respuesta temporal, en concreto a las opciones de si mostrar desplazamiento, velocidad o aceleración el desplegable de arriba y el tiempo del ensayo la opción de abajo.

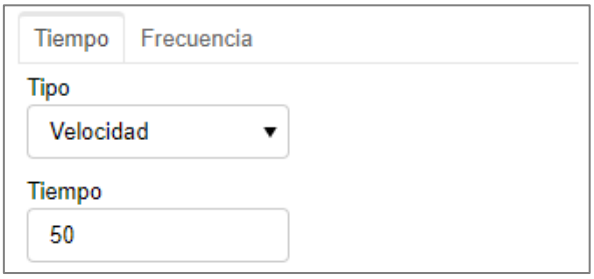


Figura 6.2: Cuadro de control sobre respuesta temporal

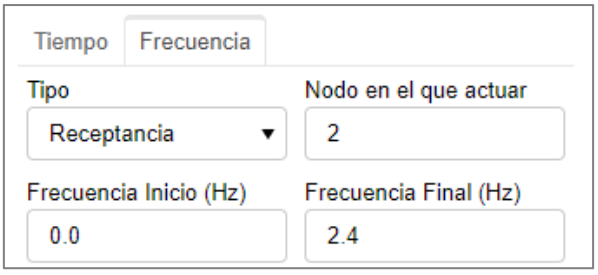


Figura 6.3: Cuadro de control sobre respuesta en frecuencia

La segunda pestaña, Frecuencia, muestra las opciones sobre la gráfica de la respuesta en frecuencia siendo estas:

- La primera de la esquina superior izquierda, un desplegable que deja elegir entre mostrar la receptancia, movilidad o acelerancia.
- La segunda de la esquina superior derecha, un contador que selecciona el grado de libertad sobre el que se actuará.
- Por último, las dos opciones muestran los valores inicial y final respectivamente del intervalo de la frecuencia.

5. **Botón de descarga de datos:** Este botón permite descargar un fichero con extensión CSV con todos los datos con los que el programa ha realizado las gráficas. Este fichero es compatible con Excel y descarga los datos tanto de la respuesta temporal como la respuesta en frecuencia.

6.4.2 Salida de datos

Una vez visto la entrada de datos, ahora se explica la segunda parte de la distribución del programa, la salida por pantalla de los resultados. Ésta dispone de tres ventanas diferente pudiendo variar para mostrar los resultados de la respuesta temporal en la pestaña de Tiempo, los resultados de la respuesta en frecuencia en la pestaña de Frecuencia y las tablas con los datos de los autovalores, autovectres y factores de amortiguamiento crítico en la pestaña de Cálculo modal.

La primera pestaña es sobre la respuesta en el tiempo:

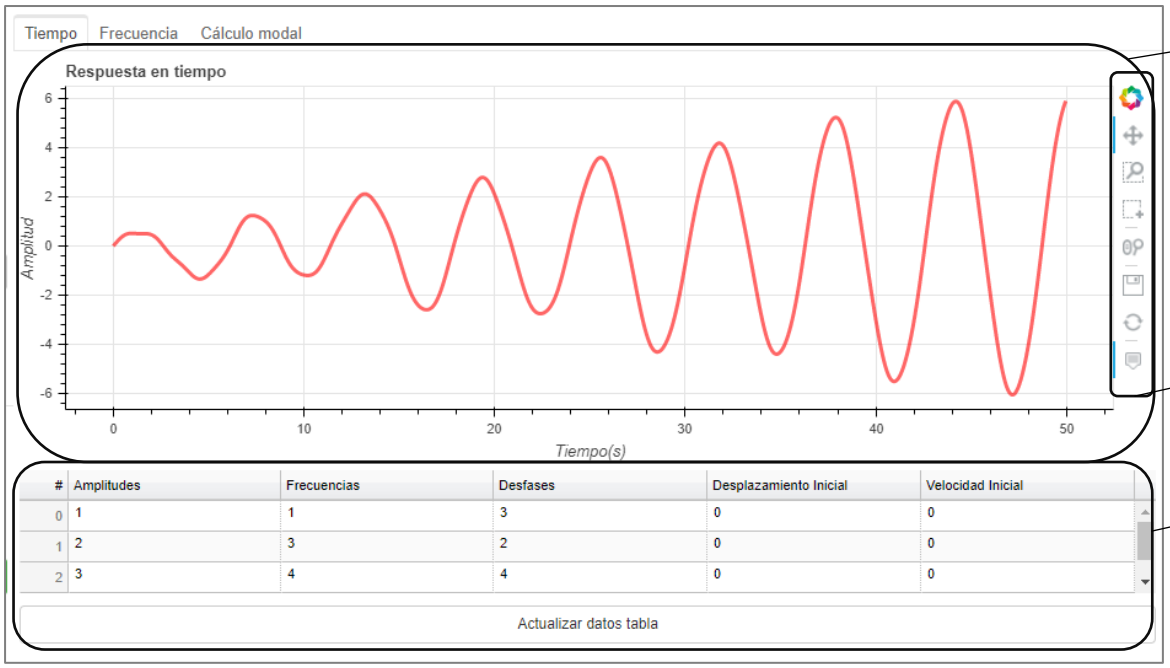


Figura 6.4: Salida de resultados de la respuesta en el tiempo

6. **Gráfica de la respuesta en el tiempo:** Esta es la gráfica de la respuesta temporal que muestra la amplitud, en las ordenadas, del grado de libertad que se ha escogido visualizar en la ventana 3 y el tiempo transcurrido, representado en las abscisas.

7. **Opciones sobre la gráfica de Bokeh:** Este menú se ha configurado para tener las herramientas esenciales. El primer logo que se muestra es el de Bokeh [3] y con él se puede acceder a la página oficial. El segundo te permite mover la gráfica libremente con el cursor. El tercero es una opción para seleccionar solo parte de la gráfica y que se centre la ventana en ella. El cuarto desactiva la opción anterior y deja fija la vista que se muestre en ese momento por pantalla. El quinto es una ruleta para hacer zoom desde la ruleta del ratón sobre la gráfica, esta opción es más precisa que la de la tercera. La sexta opción deja guardar la gráfica como se muestre en ese momento en el ordenador en un formato con extensión PNG. La séptima es un botón para restablecer la gráfica como de inicio. La última opción permite activar o desactivar que se muestren valores de la gráfica sobre un punto si se coloca el cursor sobre ella.

8. **Entrada de datos de los parámetros de las fuerzas y condiciones iniciales:** La tabla que se muestra en la ventana es parecida a la de la ventana dos de más arriba. En esta podemos variar los parámetros de las ecuaciones que definen las fuerzas que actúan sobre el sistema y las condiciones iniciales de desplazamiento y velocidad de los diferentes grados de libertad. En el caso de que las fuerzas sean nulas los valores de las tres primeras columnas serán cero, este es el caso visto en el capítulo del ensayo experimental donde solo existen desplazamientos iniciales. También dispone de un botón, debajo de la tabla, que actualiza los datos una vez se han terminado de modificar.

La segunda pestaña de esta parte, la pestaña de Frecuencia, es la que se muestra a continuación en la Figura 6.5.

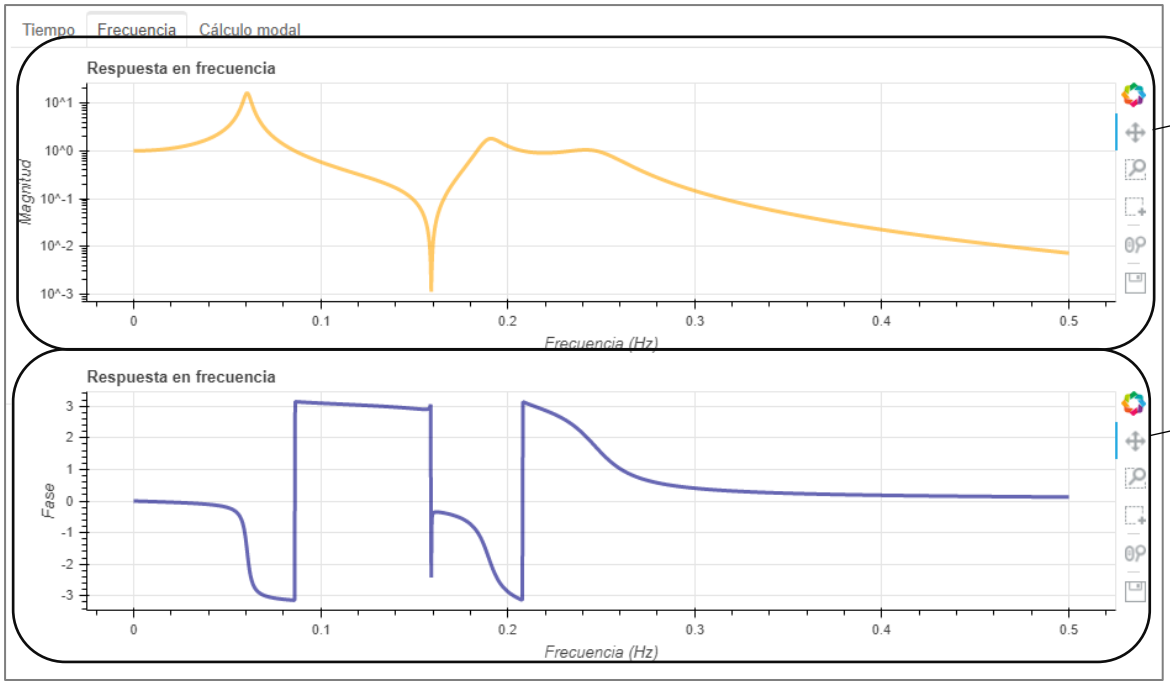


Figura 6.5: Salida de resultados de la respuesta en frecuencia

9. **Graficas de la función de respuesta en frecuencia (FRF):** Esta gráfica muestra la función de respuesta en frecuencia según los parámetros establecidos por el usuario en la ventana cuatro, en la pestaña de Frecuencia. Estas funciones siempre se grafican con la amplitud en las ordenadas y la frecuencia en las abscisas.

10. **Graficas de la fase en función de la frecuencia:** Esta gráfica, a diferencia de la anterior, muestra la fase en vez de la magnitud frente a la frecuencia.

La tercera y última ventana de la salida de datos es la de la pestaña de Cálculo modal. Esta ventana contiene cuatro diferentes tablas que proporcionan información de los datos sobre los autovalores y autovectores, tanto del caso amortiguado como del no amortiguado, y datos sobre los factores de amortiguamiento crítico.

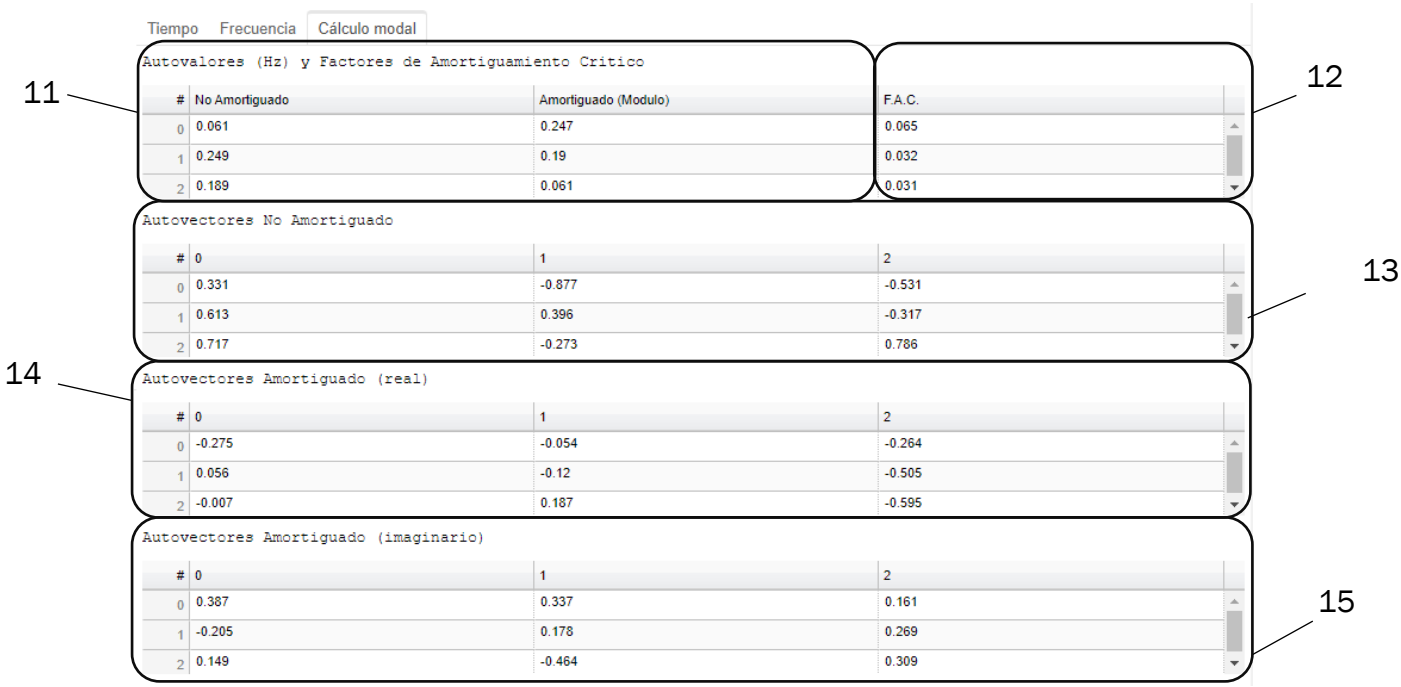


Figura 6.6: Tablas de autovalores, autovectores y factores de amortiguamiento

11. **Tabla de los autovalores:** La gráfica muestra los valores de los autovalores por filas según los grados de libertad. En la primera columna muestra los resultados del caso no amortiguado y en la segunda columna los de caso amortiguado.

12. **Tabla de los factores de amortiguamiento crítico:** Aquí se muestran los factores de amortiguamiento crítico por filas según el grado de liberad al que correspondan.

13. **Tabla de los autovectores para el caso no amortiguado:** En esta tabla el resultado es una matriz cuadrada de dimensión n. Cada columna de esta matriz es una forma modal asociada a cada una de las n frecuencias propias.

14. **Tabla de la parte real de los autovectores para el caso amortiguado:** La tabla muestra la matriz resultado, pero solo la parte real de los valores. Esto es por las limitaciones de Bokeh que no permiten mostrar números imaginarios.

15. **Tabla de la parte imaginaria de los autovectores para el caso amortiguado:** Esta es igual a la anterior, pero con su parte imaginaria.

6.5 Tareas comunes

- **Estudio del comportamiento del sistema según sus propiedades**: El programa permite visualizar la amplitud de los diferentes nodos frente al tiempo, esto es útil para observar cómo se disipa la energía o como se mantiene la gráfica constante si se suprime el amortiguamiento estudiando así su influencia.

- **Obtención de propiedades dinámicas**: Se puede modificar tanto las fuerzas que se aplican como las condiciones iniciales para estudiar el tiempo que tarda un sistema en disipar la energía o volver a su estado de reposo. Si se ensaya una estructura y se conocen esos tiempos y amplitudes, se pueden sacar propiedades dinámicas de la misma.

- **Cumplimiento de las especificaciones**: Con los datos como las propiedades dinámicas y las fuerzas exteriores se pueden obtener datos del programa para comprobar si es posible analizar la estructura y ver si cumple las especificaciones en lo que se refiere a desplazamientos o vibraciones.

- **Obtención de las Funciones de Respuesta en Frecuencia (FRF):** Se pueden descargar las gráficas de las funciones de respuesta en frecuencia y obtener de ellas las frecuencias naturales del sistema.

6.6 Funciones avanzadas

Con el fin de dar un valor añadido al programa y que destaque frente a otros similares se ha incorporado funciones más específicas. Estas funciones son las que se muestran en la ventana de Calculo modal del programa, se ven en la Figura 6.6, siendo los valores de los autovalores, autovectores y factores de amortiguamiento crítico. Por otro lado, los factores de amortiguamiento críticos son los valores de los respectivos amortiguamientos para los diferentes grados de libertad que anulan la vibración, siendo el límite entre que la estructura oscile o no. Estos valores son importantes a modo de referencia para decidir cuáles son los adecuados para la estructura según las especificaciones de la misma.

6.7 Problemas comunes

Como todo programa en su fase de desarrollo existen problemas que pueden derivar de la limitación de las herramientas usadas para la programación y la falta de experiencia en el ámbito por parte del programador. Algunos problemas se han

mostrado en las tablas de Bokeh, estas son poco modificables lo que hace difícil adaptarlas a un programa como el de este trabajo por su necesidad de adaptar tanto filas como columnas a las diferentes entradas y salidas que dependen del número de grados de libertad que es habitual que cambie para cada caso.

A continuación, se mencionan algunos de los problemas más frecuentes con los que se encontrara un usuario haciendo un uso normal del programa y soluciones a los mismos:

- **Error al introducir datos en las tablas:** Como se ha mencionado, la librería de tablas de Bokeh muestra algunas limitaciones para ser modificada. Es posible que el usuario se encuentre con que al introducir un dato nuevo en las tablas y actualizar, los valores del programa no se vean modificados. Esto sucede porque Bokeh no ha leído los datos nuevos. Es necesario que una vez se ha seleccionada una casilla de las tablas y se ha modificado, se seleccione otra casilla diferente, aunque no se modifique. De esta forma Bokeh interpreta que has dejado de modificar esa casilla y que el valor ya es definitivo.

- **Falta de sincronización de ventanas:** No ha sido posible sincronizar las ventanas de la columna de la derecha y de la izquierda para que cuando se seleccione una la otra coincida. Esto puede llevar a confusión si se están usando varias ventanas a la vez.

- **Error al descargar las gráficas:** Las gráficas se descargan con lo que se esté mostrando en ese momento por pantalla por lo que si se quiere la gráfica completa es necesario devolverla a su estado inicial.

- **El programa no responde:** Existe la posibilidad de que el programa deje de funcionar. Al ser un programa de acceso online cualquier interrupción en la red puede hacer que pierda la conexión definitivamente, aunque vuelva la conexión a la red, y sea necesario volver a cargar el buscador para que vuelva a funcionar. En estos casos se pierde el progreso, por lo que se recomienda descargar con frecuencia los datos que se deseen conservar.

- **Archivos descargados:** Los archivos descargados se descargan en formato “CSV”, es posible que el ordenador no los interprete como una extensión conocida. Por esto y por si se quieren abrir en Excel, es necesario solicitar abrir el archivo con un programa diferente y seleccionar Excel. También es posible abrir el archivo desde Excel directamente

Capítulo VII. Valoración Económica

7.1 Resumen económico

La finalidad del siguiente capítulo es resumir los costes del proyecto incluyendo el tiempo empleado y el precio de los equipos usados. Para las horas empleadas se estima el sueldo medio de un Ingeniero Mecánico recién titulado según la universidad Alfonso X el Sabio [21]. Solo se contemplan los costes directos por haberse realizado el trabajo en una modalidad telemática y no representar un coste relevante.

La Tabla 7.1 muestra la cuantía total por las horas empleadas en el desarrollo del trabajo. Estas horas incluyen las dedicadas al desarrollo del software y las dedicadas a los ensayos experimentales.

Tabla 7.1: Coste directo por mano de obra

Concepto	Cantidad
Horas semanales (hora)	10
Duración en semanas (semanas)	30
Horas totales empleadas(hora)	300
Suelda Ingeniero (€/hora)	12,5
Total (€)	3750

La segunda parte de los costes es la relacionada a los aparatos de trabajo como es el ordenador y los equipos usados en el ensayo. La Tabla 7.2 recoge el listado de estos elementos y su coste.

Tabla 7.2: Coste directo por elementos de trabajo

Concepto	Coste (€)
Portátil personal	500
Licencias de los programas	0
Maqueta ensayada	100
Sistema de adquisición Sirius DAQ	6 000
Vibrómetro Laser	1 500
Acelerómetro piezoeléctrico	350
Total	8450

La suma total de los costes directos asciende a 12200 €. Como se muestra en la Figura 7.1, donde se desglosan los costes en un gráfico, destacan los costes de los aparatos usados para la experimentación y las horas empleadas para el desarrollo del software.

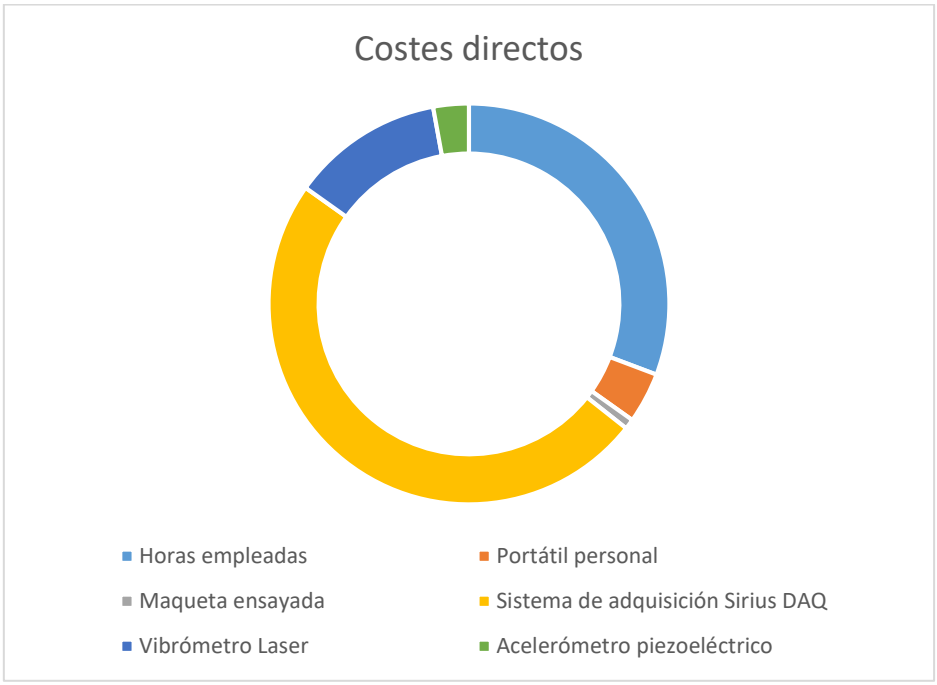


Figura 7.1: Grafica sobre los costes directos

La finalidad de este trabajo es académica y no tiene un fin lucrativo por lo que no se tiene en cuenta un posible beneficio económico y tampoco se calcula la rentabilidad del mismo y de los aparatos usados. Todos los aparatos para la experimentación y ensayo han sido proporcionados por el Departamento de Medios Continuos y Teoría de Estructuras de la Universidad de Valladolid.

Conclusiones y Líneas futuras

Conclusiones

En lo referente a las conclusiones del presente trabajo, por un lado, cumple con todos los requisitos marcados desde un inicio en cuanto a lo que englobaba el proyecto entero, desde el motor de cálculo programado en Python hasta la interfaz gráfica diseñada en la librería Bokeh. Además, supera las expectativas en cuanto a la sencillez y manejo del programa y los resultados del ensayo experimental, estos últimos requisitos no estaban definidos con claridad desde el principio, pues dependían del desarrollo del trabajo.

Por el otro lado, el trabajo ha servido para aprender en profundidad los fundamentos de la programación en Python y adquirir habilidades transversales como la resolución de problemas de forma independiente y el análisis de datos reales sobre un ensayo experimental.

De realizar de nuevo el trabajo, se optaría por estructurar el código en funciones desde un principio para facilitar el uso y tener una visión más clara. Además, se implementarían todas las soluciones y funciones que se han desarrollado y encontrado durante la realización del trabajo, por lo que el tiempo invertido en realizar un proyecto similar se reduciría. En cuanto al ensayo experimental, de realizar de nuevo el trabajo, se realizaría el mismo, aunque se optimizaría el tiempo, ya que se cuenta con la experiencia de este trabajo, para realizar el ensayo con sistemas más grandes y comprobar las limitaciones del software para implementar mejoras que lo capaciten para estructuras grandes. Para esto último también es una opción el seguimiento del programa desarrollado y la implementación de mejoras continuas.

Como valoración personal, la realización de un trabajo tan grande de una forma parcialmente independiente, sin olvidar la ayuda de los tutores, es una sensación muy satisfactoria que hace valorar de una forma diferente y muy positiva todo lo aprendido en el Grado de Ingeniería Mecánica.

Líneas futuras

Puesto que el programa ha sido diseñado para un entorno web y por tiempo no se ha podido implementar, sería lo próximo a desarrollar. Esta línea es más complicada de lo que puede parecer, pues es necesario el uso de código HTML para preparar el entorno web donde se incorpora la aplicación. Además, es necesario disponer de un servidor que almacene el programa y esté conectado a internet. El equipo debe estar preparado para un uso simultáneo debido a que el acceso es remoto y se desconoce el número de usuarios que pueden coincidir en el tiempo.

Una ampliación del presente trabajo, puede estar orientada al desarrollo del entorno web el cual se mencionaba antes. Al principio se ha mencionado el desarrollo de un entorno que contenga al programa desarrollado en este trabajo. Sin embargo, una posible línea futura es un entorno web que incluya más programas desarrollados por otros estudiantes en sus trabajos de fin de grado o máster y que sea una sola página web la que gestione todos los programas, pudiendo incluso, relacionarlos entre ellos para globalmente formar un entorno más completo.

Otra posible línea futura, puede abarcar la continuación del desarrollo del programa con nuevas funciones. El código se ha desarrollado de forma clara, ordenada y con anotaciones con el fin de facilitar el trabajo de compresión a una persona ajena a su desarrollo. Desde la implementación de nuevas funciones para la interfaz gráfica hasta funciones del programa nuevas.

Dentro del tipo de fuerzas se ha programado para introducir fuerzas senoidales sencillas, pero se podría haber programado para incluir funciones polinómicas, exponenciales de varios tipos, impulsos o aleatorias. Esto es porque son las más comunes, lo habitual es encontrarse fuerzas alternantes que pueden descomponerse en las anteriores. Además, es un tema más complejo y se ha decidido no complicar la primera versión del programa. Versiones futuras podrían incluir otros tipos de fuerzas y otras formas de introducirlas.

Referencias

[1] **Numpy**. *Numpy.org*. Accedido marzo de 2020. Obtenido de https://numpy.org/

[2] **Scipy**. *Scipy.org*. Accedido marzo de 2020. Obtenido de https://scipy.org/

[3] **Bokeh**. *Bokeh.org*. Accedido marzo de 2020. Obtenido de https://docs.bokeh.org/en/latest/docs/first_steps.html

[4] **Dash.** *Dash*. Accedido febrero de 2020. Obtenido de https://dash.plotly.com/

[5] **Scipy.linalg.** *Scipy.linalg*. Accedido junio de 2020. Obtenido de https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.eig.html

[6] **Matplotlib.** *Matplotlib*. Accedido noviebre de 2020. Obtenido de https://matplotlib.org/

[7] **Pandas.** *Pandas*. Accedido febrero de 2020. Obtenido de https://pandas.pydata.org/

[8] **Sympy.** *Sympy*. Accedido junio de 2020. Obtenido de https://www.sympy.org/en/index.html

[9] **Numpy.zeros.** *Numpy.zeros*. Accedido octubre de 2020. Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.zeros.html

[10] **Numpy.diag.** *Numpy.diag*. Accedido noviembre de 2020.Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.diag.html

[11] **Scipy.signal.StateSpace.** *Scipy.signal.StateSpace*. Accedido Diciembre de 2020 Obtenido de https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.StateSpace.html

[12] **Numpy.eye.** *Numpy.* Obtenido de https://numpy.org/devdocs/reference/generated/numpy.eye.html

[13] **Numpy.concatenate.** *Numpy.concatenate*. Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.concatenate.html

[14] **Scipy.signal.lsim.** *Scipy.signal.lsim*. Obtenido de https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.lsim.html

[15] **Numpy.arange.** *Numpy.arange*. Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.arange.html

[16] **Numpy.sqrt.** *Numpy.sqrt*. Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.sqrt.html

[17] **Numpy.ix_.** *Numpy.ix_*. Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.ix_.html

[18] **Bokeh user guide.** *Bokeh user guide*. Obtenido de https://docs.bokeh.org/en/latest/docs/user_guide/data.html

[19] **Numpy.append.** *Numpy.append*. Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.append.html

[20] **Numpy.delete.** *Numpy.delete*. Obtenido de https://numpy.org/doc/stable/reference/generated/numpy.delete.html

[21] **Alfonso Décimo el Sábio UAX.** *Alfonso X el Sábio UAX*. Obtenido de https://www.uax.com/en/blog/ingenieria/cuanto-gana-un-ingeniero-mecanico#:~:text=El%20sueldo%20de%20un%20Ingeniero%20Mec%C3%A1nico%20puede%20empezar%20en%20un,110.000%20euros%20brutos%20por%20a%C3%B1o.

Anexos

Como anexos al presente trabajo, existen varios archivos que contienen el código entero del programa almacenado en un archivo de extensión “py” perteneciente a Python. Estos se llaman “main.py” el que contiene el código principal, una carpeta denominada “analisisModal” con los archivos del código sobre las funciones implementadas, incluyendo un archivo de extensión “py” nombrado como “analisisModal.py” que contiene las líneas del código y por último un archivo llamado “export_csv.py” que contiene la parte del código encargada de descargar los datos a un archivo “csv”.