Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MÁSTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Técnicas numéricas para la detección, localización y cuantificación de daño en estructuras ligeras

Autor: D. Javier González Martín
Tutor: D. Antolín Lorenzana Ibán
Tutor: D. Álvaro Magdaleno González

Valladolid, septiembre, 2022

RESUMEN

En construcciones ligeras es relativamente sencillo obtener, tanto experimentalmente como de forma computacional por simulación, las funciones de respuesta en frecuencia (FRF). Analizando estas funciones se obtiene información detallada sobre características mecánicas de cada estructura como son las frecuencias y amortiguamientos propios, las formas modales y los factores de amplificación dinámica. Cualquier cambio estructural puede verse reflejado en las FRFs. Estos cambios pueden ser debidos a factores como envejecimiento, fisuras, roturas, etc. que se engloban en la denominación común de daño. La detección de ese daño es el objetivo de las técnicas de monitorizado estructural conocidas genéricamente como SHM (Structural Health Monitoring).

Este Trabajo Fin de Máster supone un primer acercamiento a este campo. Se trabajará a nivel computacional para comprobar que efectivamente los cambios estructurales inducidos en el modelo de simulación pueden ser detectados tras el postproceso de las FRFs.

Para dicho estudio se realiza una API utilizando MATLAB® para que haga las modificaciones pertinentes en un modelo digital de la plataforma existente en el laboratorio realizada con SAP2000® y mediante la comparación de los resultados de las distintas simulaciones conseguir determinar el daño.

PALABRAS CLAVE

SHM, Función de respuesta en frecuencia, Plataforma, API, MATLAB

ABSTRACT

In lightweight constructions it is relatively easy to obtain, both experimentally and computationally by simulation, the frequency response functions (FRF). By analysing these functions, detailed information is obtained about the mechanical characteristics of each structure such as eigenfrequencies and damping, modal shapes and dynamic amplification factors. Any structural change can be reflected in the FRFs. These changes can be due to factors such as ageing, cracks, breaks, etc. which are commonly referred to as "damage". The detection of this damage is the objective of the structural monitoring techniques known generically as SHM (Structural Health Monitoring).

This Master's Thesis is a first approach to this field. Work will be carried out at the computational level to verify that the structural changes induced in the simulation model can be detected after post-processing the FRFs.

For this study, an API is created using MATLAB® to make the appropiate modifications in a digital model of the existing platform in the laboratory using SAP2000® and by comparing the results of the different simulations, determine the damage.

KEYWORDS

SHM, Frequency response function, Platform, API, MATLAB

ÍNDICE

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN Y OBJETIVOS

1.1 Antecedentes

Las características de las estructuras van sufriendo un deterioro con el paso del tiempo a lo largo de su vida útil. A este deterioro se le conoce como envejecimiento y viene causado entre otras muchas cosas por el uso que se le da, el propio paso del tiempo, las inclemencias meteorológicas, modificaciones en el terreno o un impacto. Por su parte, los efectos del deterioro pueden aparecer de innumerables formas en las estructuras, como pueden ser la corrosión en los aceros, deformaciones o fisuras en el caso del hormigón, entre otros muchos.

La determinación del envejecimiento de las estructuras es muy importante para saber el estado de estas. Conocer la situación en la que se encuentran las estructuras permite adelantarse a posibles fallos antes de que ocurran. Esto se realiza haciendo un seguimiento de las estructuras mediante inspecciones visuales, como se verá más adelante, sin embargo, no siempre es fácil localizar el posible deterioro, ya que en muchas ocasiones este se produce en el interior de las estructuras o en zonas no visibles, por lo que resulta complicado determinar el agravamiento del envejecimiento de una estructura. A continuación, en la Tabla 1 se presentan las distintas lesiones que pueden aparecer en los elementos constructivos y las diferentes variantes asociadas a cada lesión existente:

Familia	Lesión	Variantes
Físicas	Humedades	De obra
		De capilaridad
		De filtración
		De condensación
		Accidentales
	Ensuciamiento	Por depósito
		Por lavado diferencial
	Erosión	Meteorológica
Mecánicas	Deformaciones	Desplomes
		Flechas
		Pandeos
		Alabeos
	Roturas	Grietas
		Fisuras
	Desprendimientos	Acabados continuos
		Acabados por elementos
		Elementos sueltos
	Erosión	Impactos y rozamientos

		Eólica
Químicas	Eflorescencias	Directas
		Indirectas
	Corrosión	Por oxidación previa
		Por aireación diferencial
		Por inmersión
		Por par galvánico
	Organismos	Animales
		Plantas
		Hongos
	Erosión	Pátinas
		Costras
		Lixiviación

Tabla 1 Posibles daños en los elementos constructivos [1]

Además, en la Tabla 2 pueden verse las cinco etapas que se establecen ante la aparición de un daño en la estructura.

Nivel de objetivos	Información proporcionada
Aparición de daños	Aviso de identificación de daños
Localización del daño	Localización del daño para mantenimiento
Caracterización del daño	Tipo e intensidad de las zonas dañadas
Caracterización del riesgo estructural	Riesgo/Vulnerabilidad en el estado actual
Predicción de la vida de la estructura	Estimación de la vida considerando el estado actual de la estructura

Tabla 2 Etapas del daño

La forma de averiguar el estado de las estructuras y su deterioro es mediante la realización de inspecciones a las estructuras. Suelen hacerse distintos tipos de inspecciones según el nivel de intensidad, frecuencia, medios humanos y materiales empleados para su realización. [2] Se realizan inspecciones visuales básicas de forma rutinaria para intentar detectar deterioros en etapas tempranas y poder prevenir daños más grandes. También con cierta periodicidad se realizan inspecciones principales, que consisten en inspecciones visuales más minuciosas de los elementos de la estructura. Además, se realiza una inspección en la puesta en servicio para tomarla de referencia con las siguientes. Esta ha de hacerse tras las pruebas de carga realizadas sobre la estructura según la normativa. [3] Por último, también existen las inspecciones especiales, que son aquellas que se realizan cuando se ha detectado un daño previamente o ante una situación particular. Además de realizar una inspección visual de la zona deteriorada, para llevarlas a cabo precisan de diferentes mediciones y ensayos sobre ella.

Por ello, resulta de gran importancia hallar métodos que permitan determinar el deterioro de una estructura que mediante inspecciones visuales puede pasarse por alto. Este tipo de métodos ayuda a localizar posibles daños no visibles y así poder llevar a cabo inspecciones especiales antes con las que realizar ensayos y, posteriormente, las reparaciones necesarias si así se requieren.

Las inspecciones visuales de carácter rutinario, además de no conseguir encontrar aquellos daños que no se muestran directamente en la zona estudiada, necesitan de mucho tiempo para poder realizarse en la totalidad de la estructura, por lo que los métodos que se buscan han de poder realizarse de forma rápida y sencilla evitando, en la medida de lo posible, tener que limitar el uso de las estructuras durante largos periodos de tiempo.

1.1.1 Structural Health Monitoring (SHM)

Se conoce como SHM o *Structual Health Monitoring* al campo de la ingeniería dedicado a detectar el deterioro en estructuras, su integridad y la localización de los daños basándose en el análisis de mediciones realizadas periódicamente. A través de una monitorización de la estructura en el tiempo y el historial de datos obtenidos con dichas mediciones puede determinarse el estado de la estructura. Para poder llevar a cabo métodos de este tipo es necesario el uso de sensores, la transmisión de datos y buena potencia computacional. El hecho de no solo determinar el estado, sino también ser capaz de estimar el daño en etapas prematuras permite poder reparar a tiempo o reforzar la estructura antes de que aparezca un deterioro grave y poder así alargar así su vida útil. [4, 5]

Por lo general, los métodos más utilizados en el campo de SHM son los basados en las vibraciones. Las distintas características de vibración de una estructura son función de sus parámetros físicos. Cuando existe un daño o deterioro en la estructura, dichos parámetros sufren modificaciones. Es por ello, que mediante sensores colocados en la estructura pueden analizarse las características de las vibraciones y detectar los cambios que haya podido sufrir la estructura. [6] En la Figura 1.1 se expone un esquema simplificado de la estructura general de un sistema de SHM:

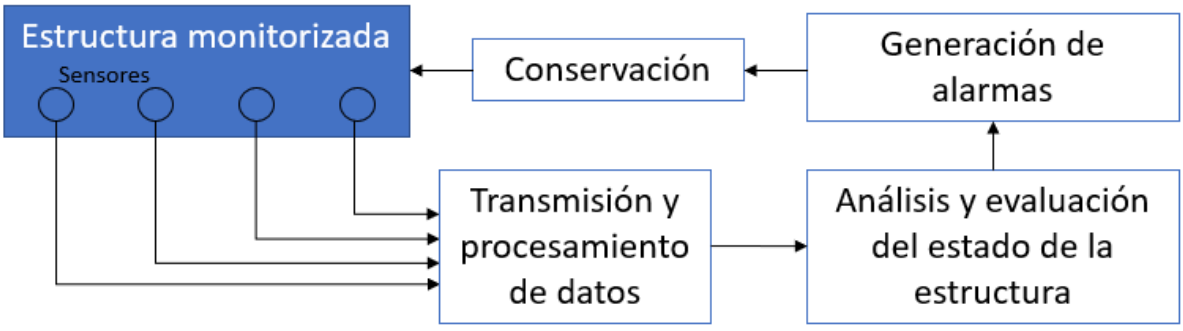

Figura 1.1 Diagrama conceptual de un sistema SHM

1.2 Metodología

En este trabajo se plantea un método para la detección de modificaciones inducidas en ciertos parámetros de estructuras a través de software basándose en los principios del SHM. El método consiste en la creación de un gemelo digital de la estructura sobre el que realizar

simulaciones con distintas modificaciones y mediante comparación de sus funciones de respuesta en frecuencia (FRF) determinar si ha podido sufrir algún tipo de variación y cuál.

Las comparaciones que se plantean para este método serán entre un modelo al que se llamará Blanco, el cual puede ser una estructura en servicio, un prototipo de laboratorio o un modelo de software entre otros, al que se le hace un análisis modal experimental (EMA) para obtener sus FRFs, y, por otro lado, un modelo Gris, que consiste en un gemelo digital de Blanco al que mediante una sucesión de simulaciones con distintas modificaciones se obtienen las distintas FRFs para compararlas con las del modelo Blanco y averiguar cuál es la causa del posible deterioro.

En este caso, al tratarse de una etapa inicial de investigación del método para determinar si es válido o no, en vez de obtener los datos con los que realizar las comparaciones con ensayos sobre una estructura o sobre un modelo a escala, como modelo Blanco se va a utilizar un modelo de software realizado con SAP2000®. Esto permitirá obtener unos resultados iniciales sobre los que establecer si el método es viable para implementarlo posteriormente en prototipos físicos y más tarde en estructuras en servicio. Además, el hecho de usar un modelo de software permite realizar las modificaciones que simulen un deterioro de una forma mucho más fácil y rápida.

El modelo que se va a utilizar es un modelo calibrado en SAP2000® de la pasarela existente en el Laboratorio S10 de la Escuela de Ingenierías Industriales (Figura 1.2), definido más adelante, sobre el que se van a aplicar distintas modificaciones de rigidez y masa que simularán algún tipo de daño sobre la estructura. Como se indica, esto se va a aplicar a una estructura modelada en SAP2000®, pero podría aplicarse a cualquier otra estructura.

Figura 1.2 Plataforma de referencia para el modelo digital

Para su realización se ha de programar una API con MATLAB® que consiga abrir los diferentes modelos de SAP2000®, modificarlos, obtener datos de ellos, calcularlos y obtener los resultados, para, dentro del propio MATLAB®, conseguir averiguar mediante comparaciones cuales son las modificaciones que se han realizado o las que más se acercan que simulan el daño de la estructura.

Como se ha dicho antes, el envejecimiento o daño puede ser de diferentes tipos como corrosión, fisuras, etc., pero en este trabajo los distintos escenarios de envejecimiento que se van a comprobar son escenarios simplificados que puedan aplicarse más adelante en la plataforma existente en el laboratorio, como lo son:

- La modificación de la rigidez de la estructura
- El aumento de la masa en un punto concreto de la estructura
- El aumento de la masa en un punto desconocido de la estructura

1.3 API MATLAB®+SAP2000®

API hace referencia a *Application Programming Interface*, es decir, una interfaz de programación de aplicaciones. Se puede definir como un mecanismo que permite la comunicación y el intercambio de información entre sistemas. [7] Mediante el uso de esta herramienta puede conectarse SAP2000® con otros softwares. Para el caso concreto de SAP2000®, permite conectarse a él a través de los principales lenguajes de programación como son VBA, Python™, C# o MATLAB® entre otros.

El uso de una API permite automatizar procesos, modificaciones y análisis a la medida. En este caso se necesita poder modificar datos, hacer cálculos y validar los resultados desde MATLAB® sin la necesidad de interactuar directamente con SAP2000®, lo que hace que la velocidad para realizar las sucesivas simulaciones sea mucho más rápida y precisa.

1.3.1 MATLAB®

Como ya se ha indicado, la API puede implementarse en distintos lenguajes de programación. En este caso se ha optado por MATLAB®. Además de como un lenguaje propio, MATLAB® se define como una plataforma de programación y cómputo numérico. Este software permite en este trabajo no solo programar el código para el correcto funcionamiento de la API realizando las distintas modificaciones necesarias y sus respectivos análisis sobre los modelos, sino también es posible realizar cálculos para comparar los resultados obtenidos en las distintas simulaciones, extraer representaciones gráficas, etc.

Para cada lenguaje, el código necesario para realizar la API varía, por lo que para la implementación de la API en MATLAB® es necesario utilizar un código específico para comunicarse con SAP2000® y ejecutar las distintas acciones que se quieran realizar sobre el modelo. En el Anexo I se encuentran descritas las diferentes acciones que se utilizan a lo largo de este TFM, así como sus definiciones y el código para implementarlas.

1.3.2 SAP2000®

Para la simulación de los distintos modelos de la pasarela y las distintas modificaciones sobre estos se utiliza el software SAP2000®. Se trata de un software de elementos finitos utilizado para el diseño de estructuras y su análisis. [8] Mediante este paquete de software es posible evaluar el análisis estático y modal de estructuras, entre otros, así como el análisis del estado

estacionario (*steady-state*), necesario en este trabajo. A través de este análisis pueden obtenerse las funciones de respuesta en frecuencia del punto requerido en cada caso en las que se basará este TFM.

Como se ha dicho anteriormente, para la simulación de la estructura se usará un modelo calibrado en SAP2000® del prototipo existente en el laboratorio del departamento. Las medidas del modelo son 13,5 m de longitud por 1 m de anchura. Además, en el centro del modelo se han incluido dos muelles con una rigidez de 19500 N/m.

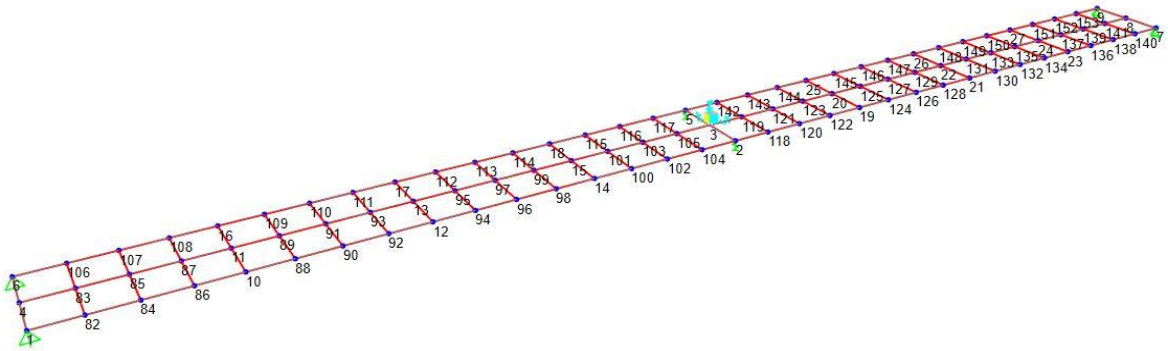

Figura 1.3 Modelado de la plataforma en SAP2000

Como puede observarse en la Figura 1.3, el modelo está compuesto de 99 puntos, aunque tienen una numeración discontinua. Esto se debe a que, al modelo inicial, el cual se compone de los nodos que van del 1 al 27 formando superficies rectangulares, posteriormente se le dividió para obtener superficies más regulares con las que obtener resultados más precisos.

En cuanto al resto de características del modelo de la pasarela, la sección utilizada consiste en una sección de tipo *shell-thick* y cuyo espesor es de 140 mm tanto para *membrane* como para *bending*. Dicha sección es de un material ortotrópico el cual se ha llamado *madera* cuyas propiedades son un módulo de elasticidad de $1{,}28 \cdot 10^{10}$ N/m^2, un coeficiente de Poisson de 0,3 y un módulo de cizalladura de $2{,}76 \cdot 10^{8}$ N/m^2. Además, la masa por unidad de volumen es de 452,38 kg.

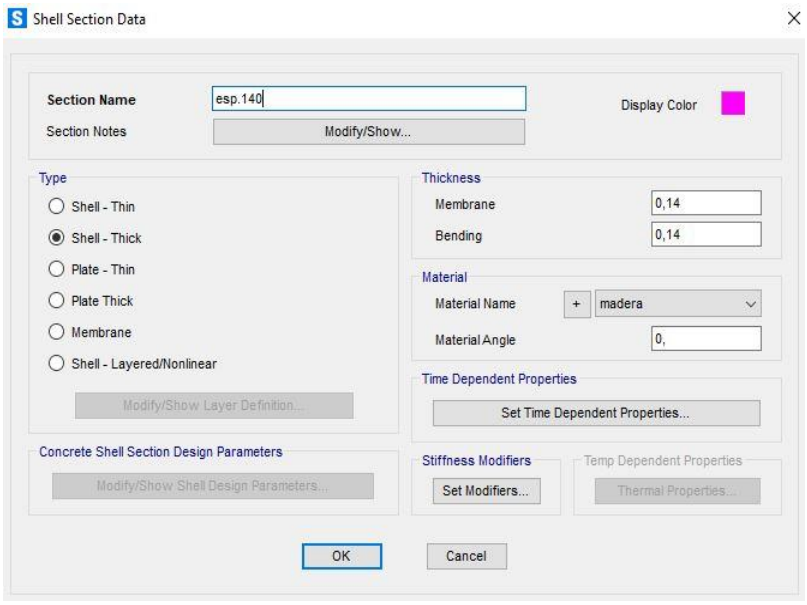

Figura 1.4 Tipo de sección y características

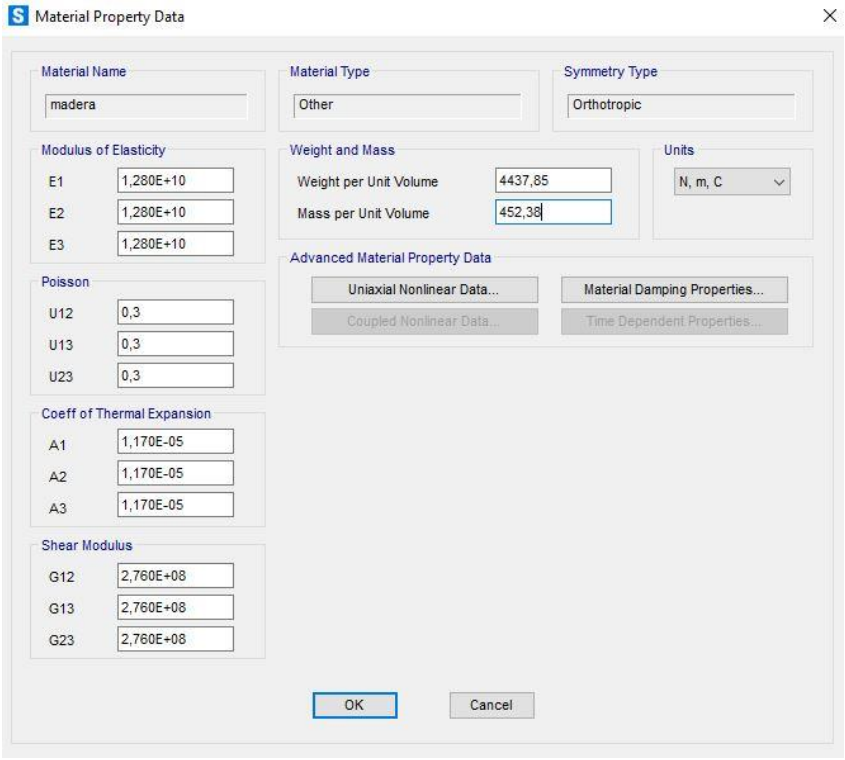

Figura 1.5 Propiedades del material

Como se ha comentado anteriormente, en este trabajo se va a hacer uso de la función *steady-state* que proporciona el programa. A través de esta función se pueden obtener las funciones de respuesta en frecuencia (FRFs). Estas funciones muestran la relación entre la respuesta de un punto de la estructura, donde se puede colocar un acelerómetro, cuando en otro punto se induce una fuerza armónica a cada frecuencia, por ejemplo, con un *shaker*. Por lo general, la respuesta es en aceleraciones. Mediante la función *steady-state* se puede simular el efecto de un *shaker* en un punto concreto de la estructura realizada mediante software y obtener los resultados.

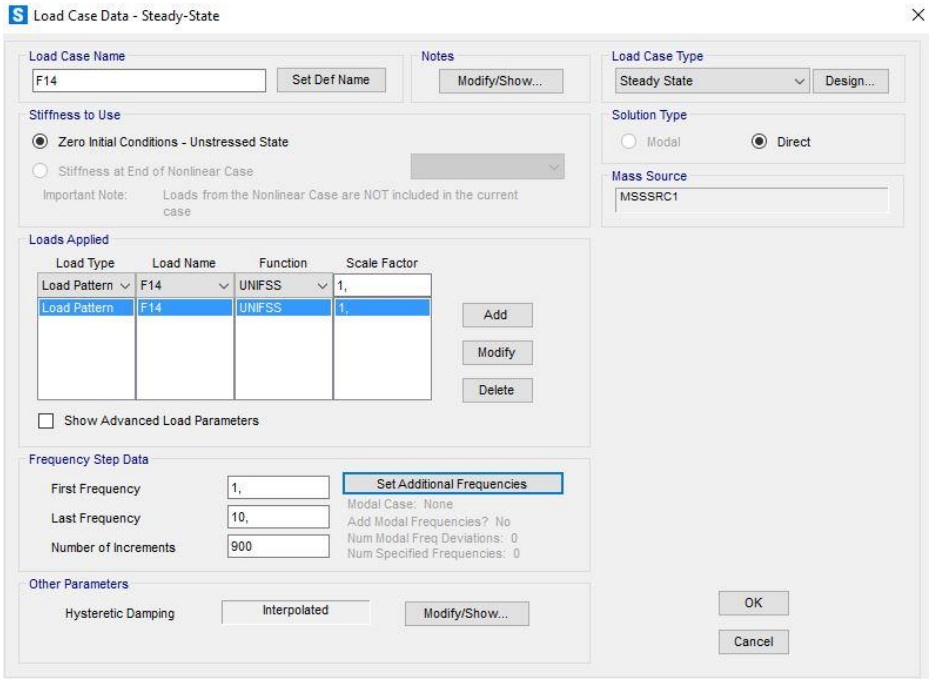

Figura 1.6 Caso de carga Steady-State

1.4 Estrategia computacional

En este punto se va a explicar el planteamiento llevado a cabo a través del software definido anteriormente para desarrollar el trabajo. Se ha dividido en tres modelos:

- Modelo Blanco: estructura estudiada de la cual se obtiene su FRF en su puesta en servicio.
- Modelo Blanco Modificado: estructura estudiada de la que se obtiene su FRF pasado un tiempo, tras sufrir un daño o después de alguna modificación de sus parámetros.
- Modelo Gris: gemelo digital de la estructura estudiada y el cual se utiliza para realizar las simulaciones y obtener las FRF para compararlas con Blanco Modificado y determinar el daño.

A continuación, se presenta una explicación más extensa de cada punto y, además, se ha realizado un esquema explicativo que puede verse en la Figura 1.7.

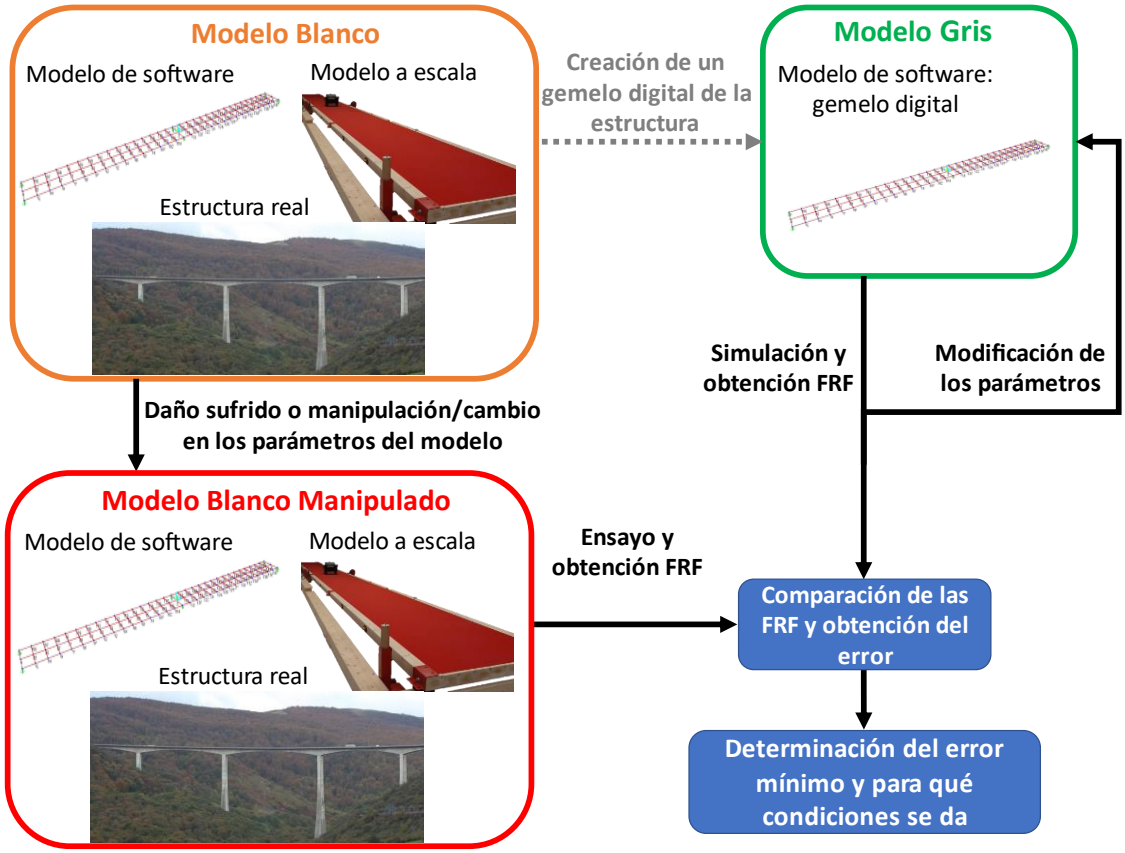

Figura 1.7 Esquema de la metodología empleada

1.4.1 Modelo Blanco

Se le llama modelo Blanco a la estructura sobre la que se quiere estudiar el envejecimiento, pudiendo ser ésta una estructura en servicio, un prototipo o modelo a escala o un modelo de software como ocurre en este caso.

Previamente a la puesta en servicio de la estructura, es necesario realizar un análisis modal experimental para obtener sus funciones de respuesta en frecuencia para, posteriormente, poder compararlas y ver si han sufrido alguna modificación, lo que podría significar un deterioro o envejecimiento.

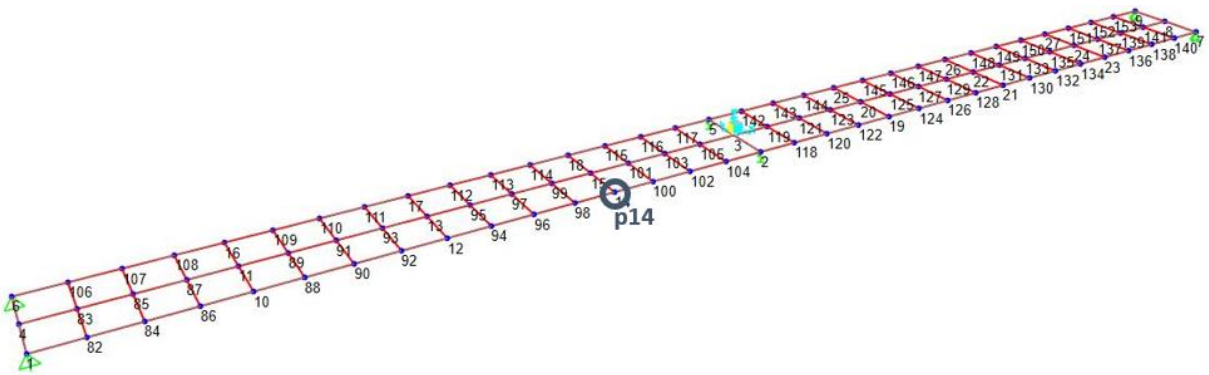

Figura 1.8 Localización del punto 14 en la plataforma

Es muy importante a la hora de simular el modelo de software, tanto en este caso como en todos los demás, tener en cuenta la masa de *shaker* que en este caso se supone de 40 kg en la posición 14 del modelo, que aparece resaltada en la Figura 1.8. Con todo ello, la gráfica de los resultados de la función de respuesta en frecuencia que se obtiene medida en el nodo 14 para el estado inicial del modelo es la siguiente:

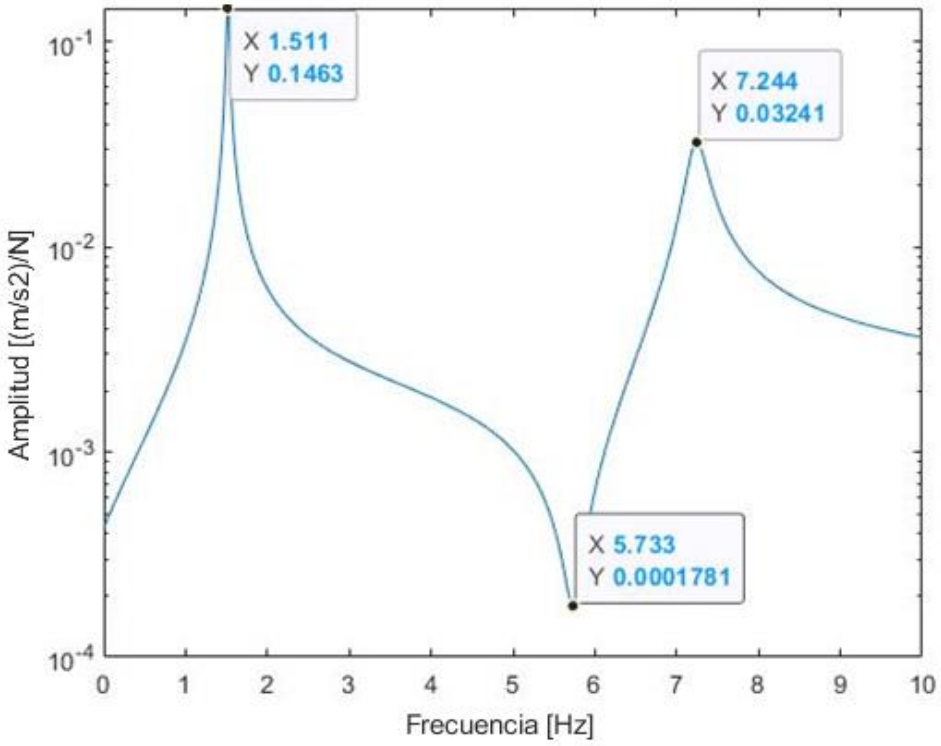

Figura 1.9 FRF de la plataforma en su puesta en servicio medida en el punto 14

En la Figura 1.9 puede verse perfectamente la gráfica de la amplitud vs. la frecuencia para el caso estudiado y del que se han destacado los puntos máximos y mínimos.

1.4.2 Modelo Blanco Manipulado

Posteriormente, una vez pasado un tiempo determinado o debido a diversas causas que hayan podido deteriorar la estructura, se ha de volver a realizar el análisis del modelo el cual pasará a considerarse Blanco Manipulado. Esta denominación viene de que, en el caso de modelos digitales, como ocurre en este trabajo, el daño vendrá de una manipulación de los parámetros de la estructura. Si la gráfica obtenida es la misma y se superpone a la anterior significará que no ha habido un deterioro significativo, en cambio, si existe alguna diferencia entre las FRF

anterior y nueva puede significar que sí que existe un envejecimiento de la estructura, así que habrá que descubrir cuál ha sido la causa de que la FRF no coincida con la anterior.

1.4.3 Modelo Gris

Independientemente de lo que se haya utilizado para el modelo Blanco (estructura, prototipo a escala, modelo de software, etc.), el modelo Gris es un gemelo digital del modelo Blanco antes de su puesta en servicio, es decir, que la FRF sea la misma que la de Blanco antes del de los daños que pueda haber sufrido la estructura. Dicho concepto de gemelo digital hace referencia a la implementación de un modelo de software de la estructura real en su estado inicial para poder ser modificada y simulada para su estudio. No obstante, en esta ocasión, al ya utilizarse un modelo de software para el modelo Blanco, el modelo Gris simplemente será una copia de éste en su estado previo a cualquier daño o manipulación de los parámetros.

Posteriormente, se irán modificando sucesivamente los parámetros pertinentes del modelo Gris y realizando su análisis para obtener la FRF y compararla con la de Blanco Modificado. Finalmente, se obtendrán las condiciones y parámetros del modelo cuyo error en la comparación de las FRFs sea mínimo.

1.5 Objetivos

Con la realización de este Trabajo de Fin de Máster se pretende alcanzar los siguientes objetivos:

- Plantear un método para analizar el deterioro de estructuras mediante sus FRF.
- Implementar dicho método mediante el uso de una API de SAP2000® con MATLAB® utilizando un modelo digital calibrado como base.
- Plantear y validar que la API funciona para un escenario en el que existe un cambio de rigidez igual en los apoyos elásticos de la estructura y cuantificar el nuevo valor.
- Plantear y validar que la API funciona para aumentos de la masa en un punto conocido de la estructura y cuantificar el aumento de dicha masa.
- Plantear y validar que la API es capaz de cuantificar y localizar en la estructura un aumento de masa en un punto desconocido de dicha estructura.

Capítulo 2. FUNDAMENTOS TEÓRICOS

En este capítulo se van a abordar los fundamentos teóricos necesarios para la realización de este TFM. En él se habla de forma resumida de los conceptos de vibración y frecuencias propias, así como del análisis modal experimental y de las funciones de respuesta en frecuencia.

2.1 Vibraciones

La vibración se define como un movimiento oscilatorio alrededor de una posición de equilibrio. La manera en la que se caracterizan las vibraciones es a través de la amplitud y de la frecuencia. Además, los sistemas poseen una frecuencia propia, también conocida como frecuencia natural. Al hacer vibrar una estructura y dejarla oscilar libremente, esta se comporta como un sistema de vibración libre amortiguada. A través del estudio de este tipo de vibración se pueden obtener tanto las frecuencias naturales como el amortiguamiento modal.

Las estructuras reales son medios continuos y como tal presentan infinitos grados de libertad y modos propios. No obstante, con fines didácticos, es interesante conocer cómo responde un sistema de un solo grado de libertad.

Cada frecuencia natural va asociada a una forma modal determinada que el modelo adopta al vibrar a dicha frecuencia. Para realizar el análisis modal experimental (EMA) de una estructura se induce una excitación externa. Como es sabido, en el caso de coincidir la frecuencia de la excitación con la frecuencia propia de la estructura se produce el fenómeno de resonancia, donde la amplitud es máxima.

2.1.1 Sistemas de un grado de libertad

Como se ha comentado, en esta explicación se va a centrarse en un sistema de un solo grado de libertad. [9] Para ello, se parte de la ecuación del movimiento del sistema (Ec. 1).

$$m\ddot{u} + c\dot{u} + ku = F(t) \tag{1}$$

Después, se divide la expresión anterior por la masa obteniendo la Ecuación 2.

$$\ddot{u} + \frac{c}{m}\dot{u} + \frac{k}{m}u = \frac{1}{m}F(t) \tag{2}$$

Se introducen en la expresión los parámetros modales de un sistema de un grado de libertad que se ven a continuación, la frecuencia natural (Ec.3) y el factor de amortiguamiento (Ec. 4).

$$\omega_0 = \sqrt{\frac{k}{m}} \tag{3}$$

$$\xi_0 = \frac{c}{2 \cdot \sqrt{k \cdot m}} \tag{4}$$

Tras la sustitución de dichos parámetros en la Ecuación 3, se obtiene la ecuación del movimiento de un sistema de un grado de libertad en función de los parámetros modales (Ec. 5).

$$\ddot{u} + 2\xi_0\omega_0\dot{u} + {\omega_0}^2 = \frac{1}{m}F(t) \quad (5)$$

Y su solución en el dominio del tiempo es la de la Ecuación 6.

$$x(t) = Ae^{-\xi\omega t} \cdot \sin(\omega t + \varphi) \quad (6)$$

Siendo:

- A: amplitud de la oscilación (metros)
- ω_0: frecuencia natural de la oscilación del cuerpo libre (rad/s)
- φ: ángulo de fase (rad)
- ξ: factor de amortiguamiento, el cual se encarga de que la vibración se suavice cuanto t→∞. Es un adimensional.

La ecuación del movimiento en función de los parámetros modales también puede expresarse en el dominio de la frecuencia a través de la transformada de Laplace para excitaciones periódicas (Ec. 7).

$$(-\omega^2 + i2\xi_0\omega_0\omega + {\omega_0}^2)Ue^{i\omega t} = \frac{1}{m}Fe^{i\omega t} \quad (7)$$

La función de respuesta en frecuencia (FRF), para cada valor de la frecuencia ω, representa la relación entre la amplitud de la respuesta estacionaria del sistema estudiado bajo la fuerza armónica F a dicha frecuencia y la amplitud de la fuerza de excitación. Se representa mediante el término $\bar{H}(\omega)$ y su magnitud, cuando la respuesta es la aceleración, es la siguiente:

$$\bar{H}(\omega) = \frac{\frac{\omega^2}{m}}{\sqrt{({\omega_n}^2 - \omega^2)^2 + (2\omega\xi\omega_n)^2}} \quad (8)$$

En la Tabla 3 pueden observarse las distintas FRFs según las magnitudes físicas que se relacionen:

Magnitud de respuesta	Tipo de excitación	FRF directa		FRF inversa	
Desplazamiento	Fuerza	Receptancia:	$\frac{U}{F} = \bar{H}(\omega)$	Rigidez dinámica:	$\frac{F}{U} = \frac{1}{\bar{H}(\omega)}$
Velocidad	Fuerza	Movilidad:	$\frac{V}{F} = i\omega \cdot \bar{H}(\omega)$	Impedancia mecánica:	$\frac{F}{V} = \frac{1}{i\omega \cdot \bar{H}(\omega)}$
Aceleración	Fuerza	Acelerancia:	$\frac{A}{F} = -\omega^2 \cdot \bar{H}(\omega)$	Masa aparente:	$\frac{F}{A} = \frac{1}{-\omega^2 \cdot \bar{H}(\omega)}$

Tabla 3 Distintas funciones de respuesta en frecuencia [9]

2.1.2 Análisis modal

El análisis modal es el proceso mediante el cual se determinan las características dinámicas inherentes de un sistema. Estas son las frecuencias naturales, los factores de amortiguamiento y las formas modales. Mediante la determinación de dichos parámetros modales, se puede evaluar la respuesta dinámica del sistema ante cualquier excitación dependiente del tiempo a través del método de superposición modal. La deflexión estática por su parte no depende del tiempo, por lo que la deformada ante cualquier caso de carga estática puede obtenerse también mediante la superposición de las formas modales, siendo estas funciones trigonométricas, mientras que las deformadas estáticas son funciones polinómicas. Por lo que, según Fourier, las deformadas estáticas pueden expresarse como suma de varias formas modales.

Desde el punto de vista práctico suele realizarse un análisis modal experimental (EMA), que busca la obtención de las propiedades modales de una estructura mediante un ensayo de comportamiento dinámico ante una fuerza de excitación conocida. Mediante un correcto EMA y estimando de forma correcta los parámetros modales, puede conseguirse un modelo de respuesta del sistema. [9]

Existen tres parámetros mediante los cuales se puede modelar modalmente el sistema físico:

- Rigidez (*K*): resistencia del sistema a ser deformado debido a una perturbación.
- Amortiguamiento (*C*): capacidad de disipación de la energía del sistema.
- Masa (*M*): inercia del sistema o resistencia que opone el sistema a sufrir aceleraciones.

Con estos parámetros y sabiendo las excitaciones a las que se somete el sistema, puede obtenerse la ecuación del movimiento del sistema dinámico (Ec. 9) al imponer como condición el equilibrio. En dicha ecuación, *K*, *C* y *M* representan las matrices de rigidez, amortiguamiento y masa respectivamente, mientras que *F(t)* representa el vector de fuerzas que se aplican sobre el sistema de forma externa.

$$M\ddot{u} + C\dot{u} + Ku = F(t) \tag{9}$$

El comportamiento dinámico de la estructura, refiriéndose a la posición, velocidad y aceleración de cualquier punto de ella, puede determinarse a través del modelo físico anterior si se conocen *M*, *C*, *K* y *F(t)*. Sin embargo, no suele ser habitual saber el valor exacto de dichos parámetros, por lo que es aquí donde cobra importancia el análisis modal. A través de él es posible modelizar el comportamiento de la estructura con distintas propiedades modales, las cuales están relacionadas a las propiedades físicas reales, y describir el comportamiento dinámico de una estructura sometida a una excitación externa sin conocer las propiedades físicas que rigen dicho comportamiento. A continuación, se presentan las distintas propiedades modales. En ellas aparece el subíndice *i* que indica el número de modo y cuyo rango va desde *i=1* a *i=N*, siendo *N* el número total de grados de libertad que existen en el sistema mecánico.

- Frecuencias naturales/propias (f_i, ω_i): son el conjunto de frecuencias a la que un cuerpo alterado de su posición de descanso tiende a vibrar libremente. Dichas frecuencias dependes únicamente de las propiedades físicas del sistema y de las condiciones de contorno. En cambio, no dependen ni de la excitación a la que se somete el cuerpo, ni a las variaciones de las condiciones iniciales que puedan ocurrir. En cuanto a su

notación, f se usa para frecuencias en Hz, mientras que ω se utiliza para aquellas en rad/s. [10]

- Modos propios (φ_i): cuando se somete una estructura a una excitación armónica de frecuencia igual a la frecuencia natural, los modos propios establecen la forma en la que dicha estructura vibra. Cada frecuencia natural lleva asociada un modo propio.
- Factores de amortiguamiento modal (ξ_i): indican el amortiguamiento asociado a cada modo de forma independiente, considerando que únicamente actúa atenuando el efecto de cada modo por separado

El EMA consiste en un ensayo en el que una estructura se excita sometiéndola a una fuerza conocida y analizar la respuesta de la estructura ante dicha fuerza. La respuesta puede ser en forma de aceleraciones obtenida en cualquier punto de la estructura. Estos ensayos se han de realizar bajo unas condiciones de control rigurosas para que los datos que se obtienen sean fieles al comportamiento dinámico de la estructura estudiada.

Figura 2.1 Esquema de la operativa para la obtención del modelo estructural

Además del análisis modal experimental, también existen otras técnicas para obtener las propiedades modales, como es el análisis modal operacional.

2.1.3 Modelado computacional y calibración

Para los objetivos que se persiguen, es necesario tener un modelo de la estructura que responda modalmente de la misma forma que lo hace la estructura real. En ingeniería estructural es usual emplear el método de los elementos finitos. Conocida la geometría, las propiedades de los materiales y las condiciones de contorno se tendría una primera versión del modelo computacional. Por indeterminaciones propias del proceso constructivo y distintas dependencias de las propiedades mecánicas de los materiales con diversos factores, suele ocurrir que las propiedades modales de esta primera versión difieran de las experimentales. No obstante, tras un proceso denominado *model updating* o calibrado computacional, se puede conseguir que el modelo responda casi como la estructura real, al menos en los primeros modos propios. Tras este proceso se llega a lo indicado en los apartados 1.3 y 1.4.

Capítulo 3. METODOLOGÍA PARA DETERMINAR CAMBIOS EN LA RIGIDEZ

El primer caso estudiado es el de determinar una reducción de la rigidez en los apoyos elásticos del modelo de la estructura. Se trata de un escenario sencillo y también viable de replicar en un futuro sobre el modelo real de la pasarela existente en el laboratorio para así poder compararlo con lo obtenido en este trabajo.

3.1 Modelo Blanco Manipulado

Con este escenario lo que se pretende es conseguir determinar la rigidez de los apoyos elásticos de un modelo los cuales se hayan podido deteriorar por el paso del tiempo. Para ello, como es evidente, es necesario un modelo manipulado, es decir, un modelo con una reducción de su rigidez. Lo que se hace es modificar el modelo Blanco creando un modelo llamado Blanco Manipulado asignando nuevas rigideces a los muelles que se encuentran en los nodos 2 y 5 señalados en la siguiente Figura 3.1.

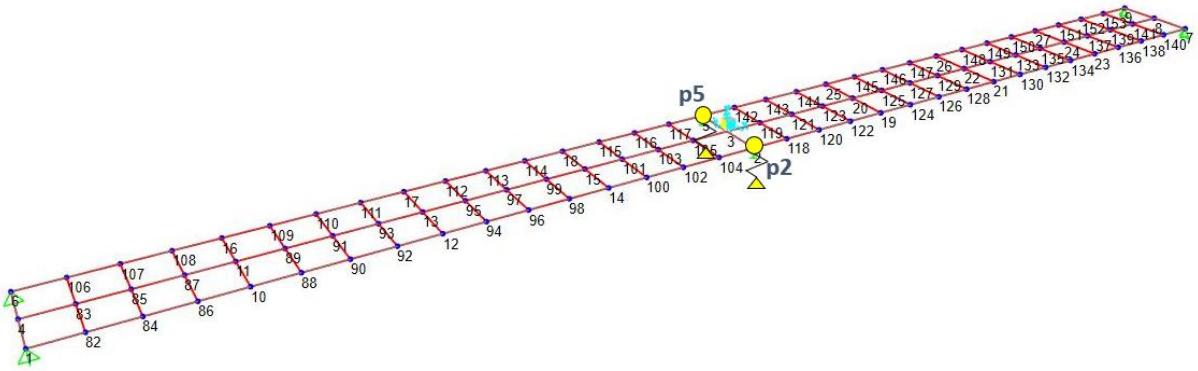

Figura 3.1 Localización de los apoyos elásticos en la plataforma

En un prototipo real esto podría realizarse intercambiando los apoyos elásticos por otros de menor rigidez. Para el caso de un modelo de software, esto se hace de una forma parecida digitalmente. En SAP2000®, una vez abierto el modelo Blanco, se seleccionan los puntos donde se encuentran los apoyos elásticos (2 y 5) a través de *Assign* > *Joint* > *Springs* y aparece la ventana de la Figura 3.2. En ella se selecciona utilizar coordenadas globales, aunque también podría hacerse con locales, y se especifica el nuevo valor de la rigidez en la traslación global Z. Además, se señala la opción de *replace existing springs* para reemplazar el valor antiguo por los anteriores que es lo que se busca.

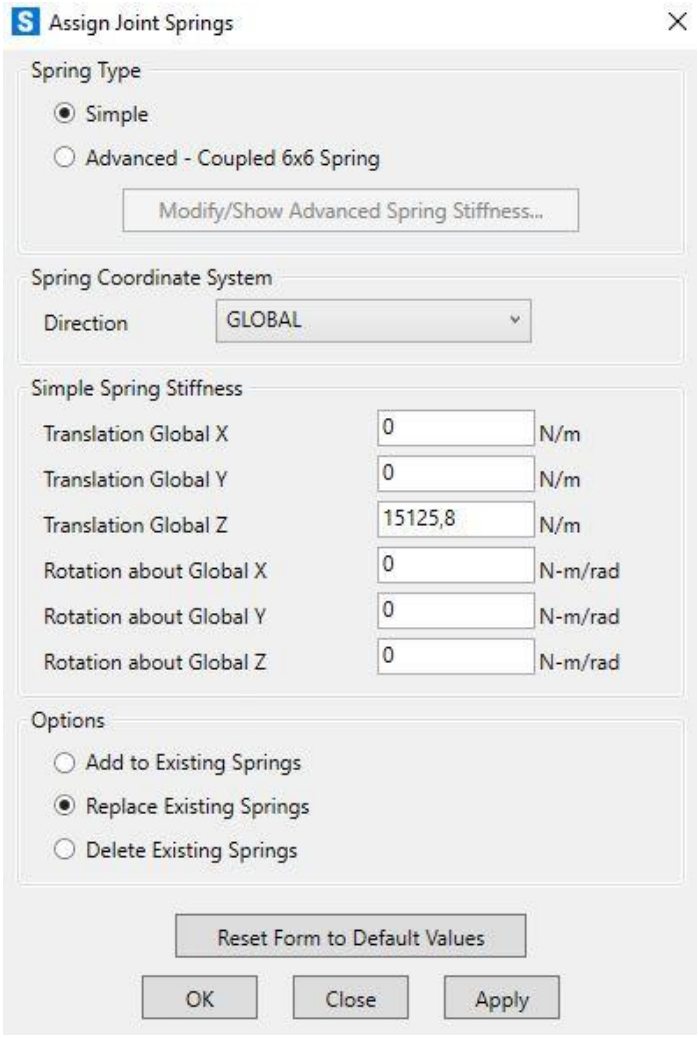

Figura 3.2 Asignación de apoyos elásticos

También se ha provisto la opción de asignar una nueva rigidez a los apoyos elásticos directamente desde el script de MATLAB®. De esta forma, con el script abierto se puede especificar el valor de las rigideces y cuando se ejecute se cambiará directamente en el modelo antes de realizar su análisis. Tras realizar dicho análisis se obtienen los valores de la función de respuesta en frecuencia los cuales MATLAB® es capaz de ofrecerlos gráficamente. Este nuevo modelo con las rigideces reducidas se considerará como Blanco Manipulado. Los datos de la FRF se tomarán en el nodo 14, como también se hizo con Blanco.

A continuación, en la Figura 3.3 se presentan gráficamente los valores de la función de respuesta en frecuencia de un ejemplo en el que la rigidez de ambos apoyos haya disminuido hasta 15125.8 N/m junto a la gráfica de la estructura en su estado inicial como se veía en la Figura 1.9. Ambas gráficas corresponden a la FRFs de la aceleración medidas en el punto 14, por lo que coinciden en dicho punto el acelerómetro y la fuerza de excitación aplicada. Esta suele denominarse FRF directa, si bien también pueden medirse las FRFs en otros puntos denominándose FRF cruzadas.

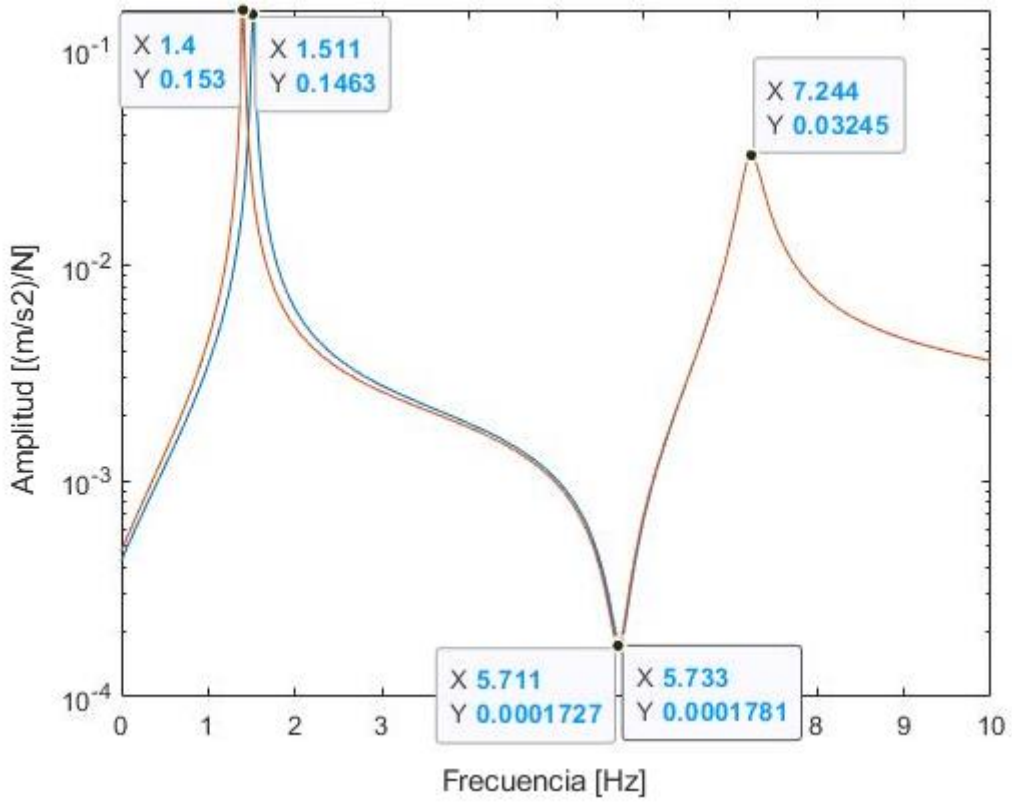

Figura 3.3 Comparación FRFs en el punto 14 para K=19500 N/m (azul) y K=15125.8 N/m (rojo)

Puede observarse que la forma de la gráfica cuya rigidez ha disminuido es muy similar a la obtenida en el modelo Blanco original que se presentaba en el Capítulo 1, sin embargo, ambas gráficas difieren ligeramente desde el inicio hasta llegar al mínimo como se ve en la Figura 3.3.

3.2 Modelo Gris

Como se explicó en el capítulo inicial, el modelo Gris es un gemelo digital del modelo Blanco, que al ser ya éste último un modelo en SAP2000®, simplemente será una copia del modelo original solo que, en este caso, en vez de asignarle unas rigideces determinadas, se irán iterando y modificando desde MATLAB® para realizar las comparaciones y cálculos pertinentes. Esto se consigue mediante un bucle *for* desde el valor más alto de la rigidez que es el del modelo original Blanco con un valor de 19500 N/m hasta 0 con un incremento determinado que se puede elegir. Este puede ser 1000, 100, 10, 1 o 0,1, pero hay que tener en cuenta que cuanto menor incremento se requiera, más tardará en calcularse.

La diferencia que se puede apreciar entre la gráfica de Blanco Manipulado y de Gris con distintos valores de rigidez tiene que ser comprobada analíticamente para establecer cuál es el error mínimo y a qué K pertenece. La forma de comparar las distintas iteraciones con el modelo Blanco Manipulado es calculando el error que existe entre ambas a lo largo de todos los puntos de la gráfica. Esto se hace con el sumatorio de las diferencias en valor absoluto de los distintos puntos de la gráfica, en este caso de 1 a 901 como se muestra a continuación:

$$\varepsilon = \sum_{j=1}^{901} \left| gris_j - blanco_j \right| \qquad (10)$$

Donde:

ε es el resultado del error

$gris_j$ es el valor en el punto j de la gráfica del modelo Gris

$blanco_j$ es el valor en el punto j de la gráfica del modelo Blanco Manipulado

Para recorrer todos los puntos de la gráfica se hace un bucle *for* de 1 hasta 901 para que pase por los 900 incrementos en los que se definió la frecuencia en la Figura 1.6. Para que la ecuación 10 sea entendida y opere correctamente dentro del código de MATLAB® ésta ha de introducirse de la siguiente forma:

$$error_j = error_{j-1} + abs(\, gris_j - blanco_j) \qquad (11)$$

La razón por la que se utiliza el valor absoluto de la diferencia de las dos gráficas se debe a que existen ocasiones en las que la gráfica simulada se encuentra por encima de la de referencia, mientras que en otras zonas se encuentra por debajo, por lo que si simplemente se tiene en cuenta la suma de las diferencias habrá valores positivos y valores negativos que en el sumatorio de todos pueden hacer que el error sea menor del que realmente es dando resultados equivocados. Esto puede verse en la Figura 3.4, que pertenece a un ejemplo de un capítulo posterior. En ella se ve como la gráfica de referencia (línea azul) se encuentra en algunas zonas por encima de la gráfica del modelo Gris (línea roja) en las que se ha marcado el área en verde, mientras que otras ocasiones se encuentra por debajo, marcado en naranja.

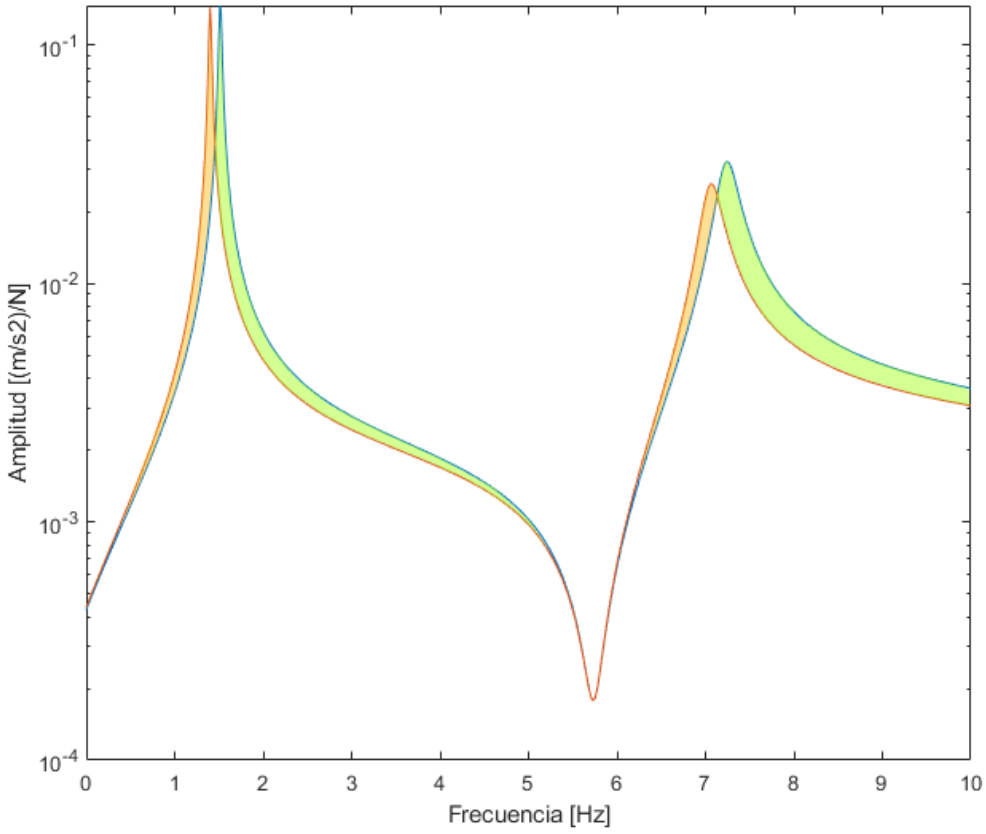

Figura 3.4 Gráfica explicativa del error positivo (naranja) y negativo (verde)

Una vez comparados los resultados del error de Blanco Manipulado con los modelos de Gris con los distintos valores de las rigideces el programa se queda con el valor de la rigidez para el cual el valor del error es menor. Esto se consigue mediante condiciones *if* en las que en caso de ser el error obtenido menor que el error mínimo que se tenía, el programa registra ese nuevo valor del error como mínimo y también lo hace con el valor de la rigidez con el que se obtiene.

Para la comprobación de este modelo se han realizado distintos experimentos. El primero consiste en que el valor de la rigidez sea común en ambos apoyos elásticos. Posteriormente, también se ha llevado a cabo el análisis de que los apoyos tengan rigideces distintas. Para esta situación existen distintos casos que se pueden dar, que solo se vea reducido un apoyo elástico, que solo se reduzca la rigidez del otro apoyo elástico o que se reduzcan ambos de forma desigual.

Para aquellos casos en los que las rigideces de ambos apoyos son distintas es necesario utilizar un doble bucle *for*. Un primer bucle que recorra las rigideces de un apoyo elástico y dentro de este otro bucle que recorra las del otro apoyo elástico. De esta forma, por cada valor de rigidez asignado al primer apoyo, se estudian las combinaciones posibles con todos los valores de rigidez en el otro apoyo, siempre teniendo en cuenta el incremento que se utilice.

3.3 Resultados

3.3.1 Misma rigidez en ambos apoyos elásticos

Gráficamente se puede observar que la diferencia de las gráficas del valor de la rigidez de Gris es mucho mayor y apreciable a simple vista para valores lejanos, pero cuanto más se acerca al valor de Blanco Manipulado menos perceptible es dicha diferencia.

Si se estudia como ejemplo un modelo Blanco Manipulado en el que ambos apoyos tienen una rigidez de, por ejemplo, 15125,8 N/m el primer análisis se hace partiendo del valor original de la rigidez que es de 19500 N/m. En ese primer caso, la diferencia es notable como se aprecia en la Figura 3.3 vista anteriormente. El pico máximo de la simulación de Gris es algo menor que el que se busca y aparece a valores ligeramente mayores de frecuencia. Además, existe una diferencia notable entre ambas gráficas desde el inicio hasta el pico mínimo.

Por su parte, para el caso de 15500 N/m que se puede ver en la Figura 3.5, a simple vista las gráficas son prácticamente idénticas, e incluso los picos máximo y mínimo solo difieren muy ligeramente tanto en valor como en la frecuencia en la que aparecen. Hay que tener en cuenta que para un incremento de 1000 N/m la rigidez más cercana a la de Blanco Manipulado es 15500 N/m ya que las otras iteraciones más cercanas serían 16500 N/m y 14500 N/m.

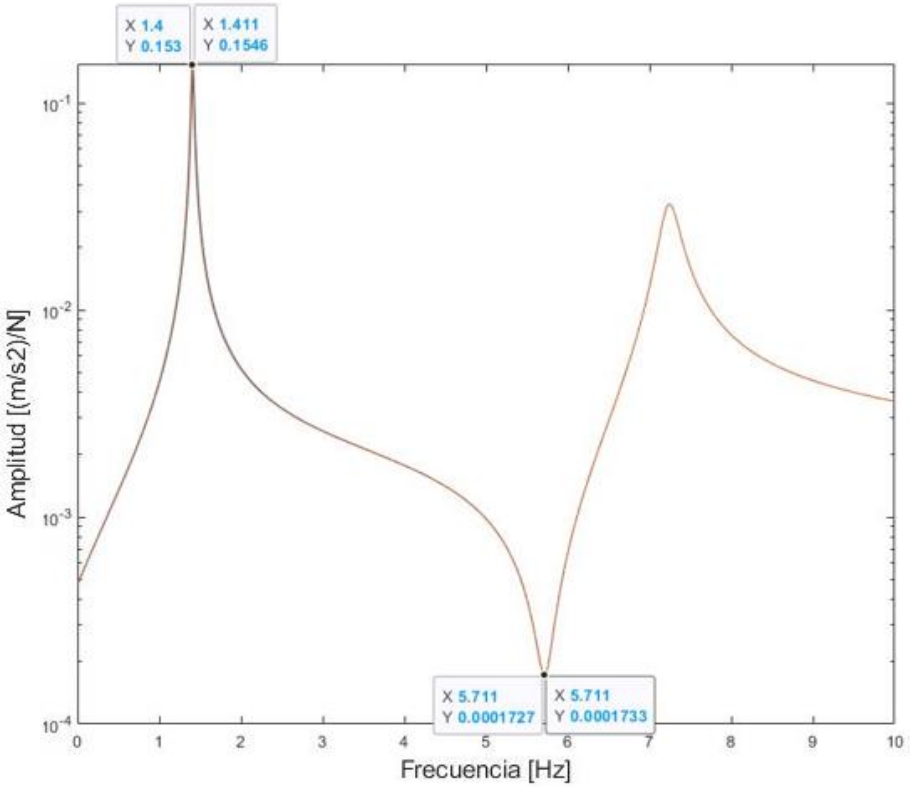

Figura 3.5 Comparación FRFs en el punto 14 para K=15500 N/m (azul) y K=15125.8 N/m (rojo)

Analizándolo gráficamente no es un buen método para ver en qué caso aparece el error mínimo, especialmente cuando los incrementos de rigidez son bajos y la diferencia entre las gráficas se hace casi imperceptible. Por este motivo, a continuación, se van a ver distintos ejemplos cuyos resultados se han obtenido numéricamente. Para estos ejemplos se han utilizado distintos valores como rigideces de los muelles del caso Blanco Modificado. Dichos valores se han elegido con el único criterio de que para obtener los resultados exactos se necesiten diferentes precisiones entre los distintos ejemplos. Por precisión se refiere al incremento elegido de la rigidez en el bucle, es decir, en algunos ejemplos se obtendrá la solución exacta reduciendo de 1000 N/m en 1000 N/m, en otros de 100 N/m en 100 N/m y así sucesivamente hasta 0,1 N/m. Además, hay que tener en cuenta que la rigidez en el bucle comienza en 19500 N/m, debido a que es la rigidez inicial de los apoyos elásticos en el modelo Blanco original, y se irá reduciendo desde ese valor según la precisión

Para un incremento de 1000 N/m en el bucle, como se puede ver en la Tabla 4, se observa que los errores relativos de las rigideces son bastante bajos. Por su parte, el error entre los valores de las FRF es también muy reducido, siendo menores cuanto más cerca del resultado de la rigidez hasta llegar a 0 cuando se alcanza la K exacta como en el caso de 17500 N/m.

K Blanco Modificado (N/m)	Incremento (N/m)	K Gris obtenido (N/m)	Error FRF [Gri-Bla]	Error relativo (%)
19000,00	1000	19500	0,3348	2,63%
18700,00	1000	18500	0,1418	1,07%
18350,00	1000	18500	0,1015	0,82%
17994,00	1000	17500	0,3476	2,75%
17500,00	1000	17500	0	0,00%

16700,00	1000	16500	0,1458	1,20%
16486,00	1000	16500	0,0104	0,08%
15125,80	1000	15500	0,2748	2,47%
14506,90	1000	14500	0,0052	0,05%
13222,20	1000	13500	0,1989	2,10%

Tabla 4 Resultados del error para distintos casos de rigidez con un intervalo de 1000 N/m

Cuando se reduce el incremento a 100 N/m (Tabla 3), se ve como tanto el error relativo de las rigideces, como el error entre las FRF se reducen en la mayoría de los casos, salvo aquellos en los que la rigidez obtenida en Gris es la misma que en el caso anterior. Además, aumentan el número de casos para los que el error es 0.

K Blanco Modificado (N/m)	Incremento (N/m)	K Gris obtenido (N/m)	Error FRF [Gri-Bla]	Error relativo (%)
19000,00	100	19000	0	0,00%
18700,00	100	18700	0	0,00%
18350,00	100	18400	0,0348	0,27%
17994,00	100	18000	0,0041	0,03%
17500,00	100	17500	0	0,00%
16700,00	100	16700	0	0,00%
16486,00	100	16500	0,0104	0,08%
15125,80	100	15100	0,0190	0,17%
14506,90	100	14500	0,0052	0,05%
13222,20	100	13200	0,0172	0,17%

Tabla 5 Resultados del error para distintos casos de rigidez con un intervalo de 100 N/m

Reduciendo el incremento a 10 N/m (Tabla 2) el modelo aumenta su precisión y se consigue reducir el error tanto absoluto entre las FRF como el relativo entre las rigideces, que pasan a ser inferiores al 0,04%. Si bien el tiempo de análisis que es necesario se ve aumentado notablemente ya que es necesario simular 100 veces más rigideces que para un intervalo de 1000 N/m. Esto hace que prácticamente solo sea posible usarlo reduciendo los intervalos del bucle.

K Blanco Modificado (N/m)	Incremento (N/m)	K Gris obtenido (N/m)	Error FRF [Gri-Bla]	Error relativo (%)
19000,00	10	19000	0	0,00%
18700,00	10	18700	0	0,00%
18350,00	10	18350	0	0,00%
17994,00	10	17990	0,0027	0,02%
17500,00	10	17500	0	0,00%
16700,00	10	16700	0	0,00%
16486,00	10	16490	0,0030	0,02%
15125,80	10	15130	0,0031	0,03%
14506,90	10	14510	0,0023	0,02%
13222,20	10	13220	0,0017	0,02%

Tabla 6 Resultados del error para distintos casos de rigidez con un intervalo de 10 N/m

Para un incremento de 1 (Tabla 7), solo aparecen errores en aquellas rigideces que precisan de decimales, siendo dichos errores mucho más pequeños que para los incrementos anteriores e imperceptibles si se analizan gráficamente. Para todas las demás rigideces sus errores obtenidos son nulos. Para hacerlo viable en cuanto al tiempo de cálculo es necesario reducir también los intervalos entre los que se buscan las rigideces.

K Blanco Modificado (N/m)	Incremento (N/m)	K Gris obtenido (N/m)	Error FRF [Gri-Bla]	Error relativo (%)
19000,00	1	19000	0	0,00%
18700,00	1	18700	0	0,00%
18350,00	1	18350	0	0,00%
17994,00	1	17994	0	0,00%
17500,00	1	17500	0	0,00%
16700,00	1	16700	0	0,00%
16486,00	1	16486	0	0,00%
15125,80	1	15126	1,48E-04	0,0013%
14506,90	1	14507	7,49E-05	0,0007%
13222,20	1	13222	1,54E-04	0,0015%

Tabla 7 Resultados del error para distintos casos de rigidez con un intervalo de 1 N/m

Finalmente, en la Tabla 8 se ve los resultados para un incremento de 0,1, teniendo en cuenta que todas las rigideces estudiadas como mucho contenían un decimal, el error en todas ellas es 0. En este caso el intervalo a realizar ha de ser muy reducido ya que, por cada unidad de rigidez, se han de hacer 10 iteraciones con sus respectivos análisis, cálculos y comparaciones.

K Blanco Modificado (N/m)	Incremento (N/m)	K Gris obtenido (N/m)	Error FRF [Gri-Bla]	Error relativo (%)
19000,00	0,1	19000	0	0,00%
18700,00	0,1	18700	0	0,00%
18350,00	0,1	18350	0	0,00%
17994,00	0,1	17994	0	0,00%
17500,00	0,1	17500	0	0,00%
16700,00	0,1	16700	0	0,00%
16486,00	0,1	16486	0	0,00%
15125,80	0,1	15125,8	0	0,00%
14506,90	0,1	14506,9	0	0,00%
13222,20	0,1	13222,2	0	0,00%

Tabla 8 Resultados del error para distintos casos de rigidez con un intervalo de 0.1 N/m

El hecho de que ambos modelos, Blanco Manipulado y Gris, sean ambos modelos de SAP2000® permite que los resultados sean exactos al comparar ambos modelos sin que estos se vean alterados por condiciones externas como ruido y el resultado del error obtenido pueda ser 0.

3.3.2 Distinta rigidez en ambos apoyos elásticos

También se ha estudiado la posibilidad de que debido al deterioro la rigidez no sea la misma en ambos apoyos. Como ya se ha indicado anteriormente, esto puede darse de varias formas, que solo se vea reducida la rigidez de un apoyo, que solo se reduzca en el apoyo contrario o que lo haga en ambos de forma desigual.

En primer lugar, dentro de este subapartado, se va a estudiar la posibilidad de que únicamente se vea reducida en uno de los dos apoyos, bien sea el apoyo elástico que se encuentra en el punto 2 o el del lado contrario en el punto 5 marcados en la Figura 3.1. Esto podría ocurrir en una estructura real en el caso de que se descalzara uno de los apoyos o se asentara, por ejemplo.

Como puede verse a continuación, al analizar un ejemplo en el que el primer apoyo (K2) se deteriore y vea su rigidez reducida hasta 16250 N/m, manteniéndose el otro apoyo (K5) con la misma rigidez inicial. Los resultados con los que se obtienen los errores mínimos son los de la Tabla 9 para los distintos incrementos vistos anteriormente (1000, 100, 10, 0 y 0.1 N/m). Se han reflejado únicamente aquellos casos en los que el error es mínimo, dejándose de lado el resto de las combinaciones.

Hay que señalar que se han resaltado en verde los casos en los que el error mínimo se da en la combinación que más se acerca a la real, mientras que en rojo se han resaltado aquellas combinaciones con las que se obtiene el error mínimo, pero no son las más cercana a las rigideces de Blanco Manipulado.

Caso K2=16250 N/m y K5=19500 N/m			
Incremento (N/m)	**K2 (N/m)**	**K5 (N/m)**	**Error FRF [Gri-Bla]**
1000	**16500**	**19500**	**0,087697**
	19500	16500	0,087698
100	**16300**	**19500**	**0,017919**
	19500	16300	0,017949
10	**16250**	**19500**	**0**
	19500	16250	2,79e-6
1	**16250**	**19500**	**0**
	19500	16250	2,79e-6
0,1	**16250**	**19500**	**0**
	19500	16250	2,79e-6

Tabla 9 Resultados del error en el caso K2=16250 N/m y K5=19500 N/m distintos intervalos

En este caso puede observarse que, independientemente de la precisión, en todos los casos el resultado con el error mínimo se da para la combinación más cercana a la real. Sin embargo, al estudiarse el caso opuesto en el que las rigideces están intercambiadas, es decir, K2=19500 N/m y K5=16250N/m como se ve en la Tabla 10, el resultado no es el mismo. Sí que se obtiene como combinaciones con error mínimo las de las rigideces más cercanas a las que se buscan, pero no en todos los casos asignadas correctamente, es decir, en los dos primeros casos para el incremento de 1000 N/m y el de 100 N/m el error mínimo aparece para el caso contrario, siendo el apoyo elástico con la rigidez reducida el del punto 2 en vez de el del punto 5.

Caso K2=19500 N/m y K5=16250 N/m			
Incremento (N/m)	**K2 (N/m)**	**K5 (N/m)**	**Error FRF [Gri-Bla]**
1000	**16500**	**19500**	**0,087696**
	19500	16500	0,087697
100	**16300**	**19500**	**0,017918**
	19500	16300	0,017919
10	16250	19500	2,79e-6
	19500	**16250**	**0**
1	16250	19500	2,79e-6
	19500	**16250**	**0**
0,1	16250	19500	2,79e-6
	19500	**16250**	**0**

Tabla 10 Resultados del error en el caso K2=19500 N/m y K5=16250 N/m distintos intervalos

En cambio, para los incrementos de 10, 1 y 0.1 N/m sí que se encuentra la combinación correcta y además exacta, ya que con dichas precisiones si es que posible alcanzar el valor de las rigideces idénticas a las de Blanco Manipulado.

Además, como en el caso del subapartado anterior, la elección de incrementos más bajos supone conseguir mejores precisiones y resultados más exactos, sin embargo, también implican una gran diferencia en los tiempos de análisis de la estructura siendo muy desfavorables para incrementos muy bajos como 1 N/m o 0,1 N/m. De hecho, al tratarse de un doble bucle *for*, cada análisis resulta mucho más prolongado que en los casos anteriores.

Al analizar otro ejemplo en el que sea necesario usar precisiones más pequeñas para encontrar el error 0, en este caso es K2 la que se deteriora hasta 17321,6 N/m, se puede observar que ocurre lo mismo que en el caso anterior obteniéndose errores mínimos muy similares para la combinación de rigideces más cercana a la exacta y para el caso contrario, y, al igual que antes, no siempre se obtiene el mínimo en la combinación más cercana a la real de Blanco Manipulado, sino en la combinación opuesta.

Caso K2=17321,6 N/m y K5=19500 N/m			
Incremento (N/m)	**K2 (N/m)**	**K5 (N/m)**	**Error FRF [Gri-Bla]**
1000	**17500**	**19500**	**0,060091**
	19500	17500	0,060091
100	17300	19500	0,007402
	19500	**17300**	**0,007401**
10	17320	19500	5,47e-4
	19500	**17320**	**5,46e-4**
1	**17322**	**19500**	**1,37e-4**
	19500	17322	1,37e-4
0,1	**17321,6**	**19500**	**0**
	19500	17321,6	1,87e-6

Tabla 11 Resultados del error en el caso K2=17321.6 N/m y K5=19500 N/m distintos intervalos

Estudiando el caso opuesto (Tabla 12), es decir, que la rigidez del punto 2 no se vea alterada K2 = 19500 N/m, mientras que la del punto 5 se reduce hasta 17321,6 N/m, e igualmente medidas las FRF desde el punto 14 se obtienen datos muy similares a los anteriores.

Caso K2=19500 N/m y K5=17321,6 N/m			
Incremento (N/m)	**K2 (N/m)**	**K5 (N/m)**	**Error FRF [Gri-Bla]**
1000	**17500**	**19500**	**0,060090**
	19500	17500	0,060091
100	17300	19500	0,007402
	19500	**17300**	**0,007401**
10	**17320**	**19500**	**5,47e-4**
	19500	17320	5,47e-4
1	**17322**	**19500**	**1,36e-4**
	19500	17322	1,37e-4
0,1	17321,6	19500	1,87e-6
	19500	**17321,6**	**0**

Tabla 12 Resultados del error en el caso K2=19500 N/m y K5=17321.6 N/m distintos intervalos

También hay que señalar que pese a obtener un error mínimo, los de resultados de combinaciones cercanas utilizando incrementos bajos como son 1 N/m o 0,1 N/m dan errores muy cercanos al mínimo que al implementarse en prototipos físicos podrían llevar a error.

Por último, la otra posibilidad que se ha estudiado dentro de que tengan ambos apoyos rigideces distintas es que ambos se hayan deteriorado. Esto implica que la rigidez sea menor de 19500 N/m en los dos apoyos elásticos. A continuación, se muestran distintos ejemplos realizados utilizando únicamente el incremento de 1000 N/m. Se ha señalado en cada tabla el resultado del valor mínimo, siendo verde en caso de que coincida con el valor exacto de las rigideces o el más cercano, y en rojo en caso de que sea otro.

Caso K2=13720 N/m y K5=16250 N/m		
K2 (N/m)	**K5 (N/m)**	**Error FRF [Gri-Bla]**
19500	10500	0,0103571
18500	11500	0,0105447
17500	12500	0,0106850
16500	13500	0,0107791
15500	14500	0,0108259
14500	15500	0,0108258
13500	16500	0,0107787
12500	17500	0,0106847
11500	18500	0,0105437
10500	**19500**	**0,0103558**

Tabla 13 Resultados del error en el caso K2=13270 N/m y K5=16250 N/m con intervalo 1000 N/m

Caso K2=18620 N/m y K5=16130 N/m		
K2 (N/m)	**K5 (N/m)**	**Error FRF [Gri-Bla]**
19500	16500	0,0466777
18500	17500	0,0467220
17500	18500	0,0467218
16500	**19500**	**0,0466772**

Tabla 14 Resultados del error en el caso K2=18620 N/m y K5=16250 N/m con intervalo 1000 N/m

Caso K2=18365 N/m y K5=17500 N/m		
K2 (N/m)	**K5 (N/m)**	**Error FRF [Gri-Bla]**
19500	15500	0,0905657
18500	16500	0,0906353
17500	17500	0,0906584
16500	18500	0,0906349
15500	**19500**	**0,0905650**

Tabla 15 Resultados del error en el caso K2=18365 N/m y K5=17500 N/m con intervalo 1000 N/m

Caso K2=13500 N/m y K5=18500 N/m		
K2 (N/m)	**K5 (N/m)**	**Error FRF [Gri-Bla]**
19500	12500	1,44e-4
18500	13500	4,36e-6
17500	14500	9,85e-5
16500	15500	1,46e-4
15500	16500	1,46e-4
14500	17500	9,72e-5
13500	**18500**	**0**
12500	19500	1,46e-4

Tabla 16 Resultados del error en el caso K2=13500 N/m y K5=18500 N/m con intervalo 1000 N/m

En todos ellos puede verse como se obtienen errores muy bajos y similares para combinaciones muy dispares. Dichas combinaciones guardan una similitud todas ellas y es que la suma de ambas rigideces es la más cercana a la suma de las dos rigideces del modelo real de Blanco Manipulado. Además, salvo cuando el error da 0 N/m, debido a que con el incremento de 1000 N/m se pasa durante el análisis por el caso de la combinación de rigideces exacta, en el resto de los casos el error mínimo obtenido no se da en la combinación más similar a la real.

Debido a esto de ha decidido ver cuáles son los resultados que se obtienen si se buscan las funciones de respuesta en frecuencia en otros puntos además de en el nodo 14. En este caso se han usado las FRF de los puntos 10 y 12 señalados en la Figura 3.6.

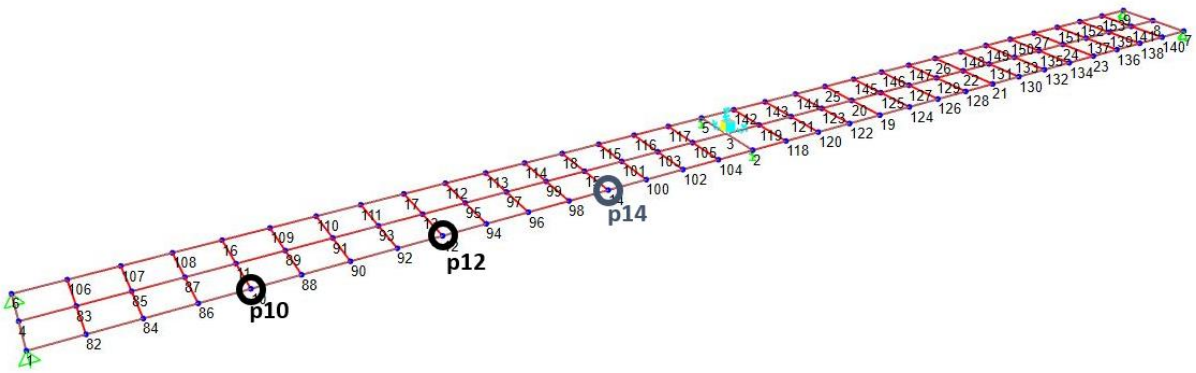

Figura 3.6 Localización de los puntos 10, 12 y 14 en la plataforma

Como se observa en la Tabla 3.7 con el primer caso estudiado anteriormente en el que K2=13720 N/m y K5=16250 N/m, siguen encontrándose los mismos problemas para las FRF de los nodos 10 y 12. Por lo que de esta forma no se consigue llegar a resultados correctos de la rigidez de ambos apoyos cuando en ambos casos están deteriorados. Esto ocurre esencialmente por la simetría, no siendo posible obtener las combinaciones de rigideces correctas.

Caso K2=13720 N/m y K5=16250 N/m				
K2 (N/m)	K5 (N/m)	Error FRF [Gri-Bla]		
		FRF p14	FRF p10	FRF p12
19500	10500	0,0103571	0,0043845	0,0080282
18500	11500	0,0105447	0,0044639	0,0081735
17500	12500	0,0106850	0,0045235	0,0082825
16500	13500	0,0107791	0,0045631	0,0083550
15500	14500	0,0108259	0,0045829	0,0083912
14500	15500	0,0108258	0,0045828	0,0083910
13500	16500	0,0107787	0,0045628	0,0083544
12500	17500	0,0106847	0,0045230	0,0082814
11500	18500	0,0105437	0,0044633	0,0081721
10500	**19500**	**0,0103558**	**0,0043837**	**0,0080263**

Tabla 17 Resultados del error para el caso K2=13720 N/m y K5=16250 N/m con intervalo 1000 N/m para las FRF medidas en los puntos 10, 12 y 14

3.3.3 Resumen de los resultados obtenidos

Tras el análisis de los resultados, para el caso que se planteaba en los objetivos, en el cual el deterioro de la rigidez es común en ambos apoyos elásticos el error mínimo se consigue para el caso de rigidez más cercano al que se busca alcanzándose el valor exacto cuando la precisión lo permite según el caso.

Por su parte, cuando la rigidez no es común en ambos apoyos los resultados no son tan claros. Si solo se reduce en uno de ellos, analizando los resultados se ve que el error mínimo se obtiene paras las combinaciones de valores de rigideces más cercanas a la que se buscan, pero no siempre se obtiene correctamente el apoyo al que pertenece cada rigidez, sino que en ocasiones se obtiene como error mínimo el caso en el que los valores de las rigideces aparecen intercambiados con respecto a los que se buscan.

Por último, en el estudio del caso en el que ambos apoyos reducen su rigidez de forma distinta, se ha visto que no se consiguen resultados aceptables debido a la simetría.

Capítulo 4. METODOLOGÍA PARA DETERMINAR LA VARIACIÓN DE MASA EN UN PUNTO CONCRETO

El siguiente escenario que se comprueba es el de conseguir averiguar el valor de una masa que se encuentra en un punto conocido de la estructura. Este modelo es fácilmente implementable más adelante en un prototipo real simplemente añadiendo una masa en el punto elegido.

El lugar elegido sobre el que aplicar la masa es el punto 14 del modelo que es al mismo tiempo donde se aplica el *shaker* y también donde se recogen los datos de la FRF.

4.1 Modelo Blanco Manipulado

En esta ocasión el modelo Blanco Manipulado es un modelo con una masa añadida en el punto 14, que se encuentra a 3L/8 del extremo de la estructura. Para realizar esto en SAP2000®, una vez abierto el modelo original, se selecciona el punto 14, y se selecciona *Assign* > *Joint* > *Masses* y se añade la masa en las tres direcciones como se muestra en la Figura 4.1. Mejor utilizar la opción de reemplazar por si hubiera masas añadidas previamente.

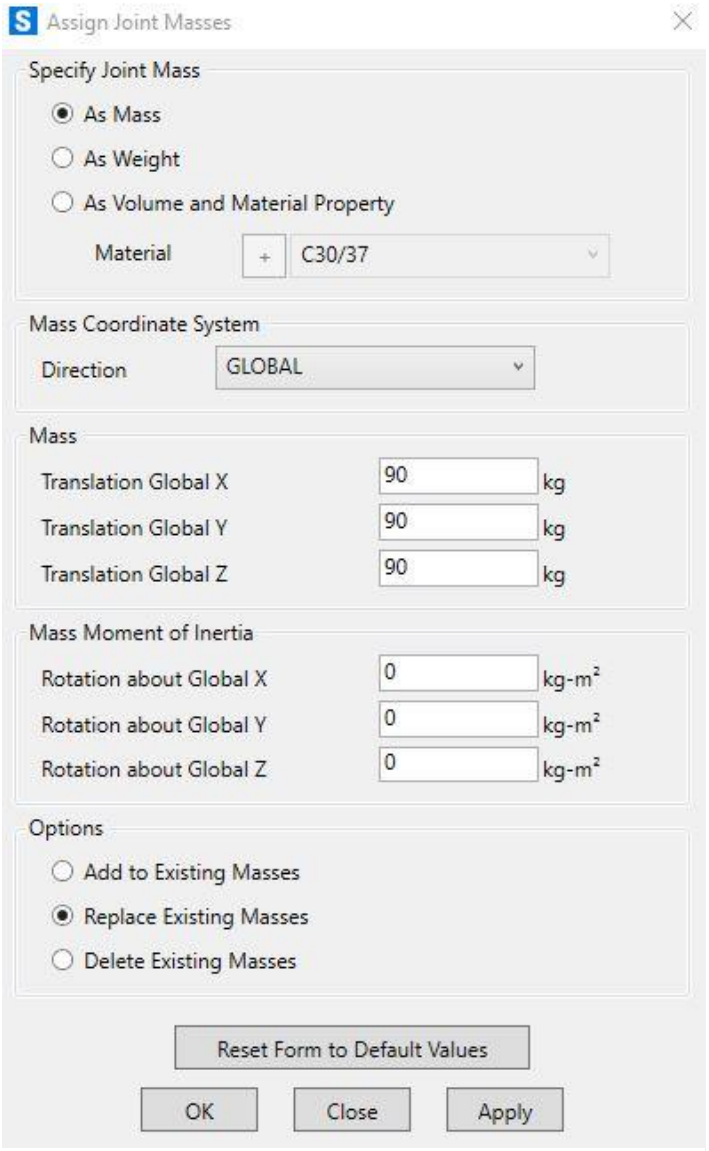

Figura 4.1 Asignación de una masa en un punto

Es importante tener en cuenta que el punto donde se aplica dicha masa es también donde se posiciona el *shaker*, por lo que es imprescindible que a la masa que se quiera establecer sobre el punto se le añada la masa asociada al dispositivo de excitación que es de 40 kg.

Realizando la simulación para un caso en el que la masa añadida total sea de 90 kg (50 kg más 40 kg del *shaker*) en el punto 14 se obtiene la siguiente gráfica:

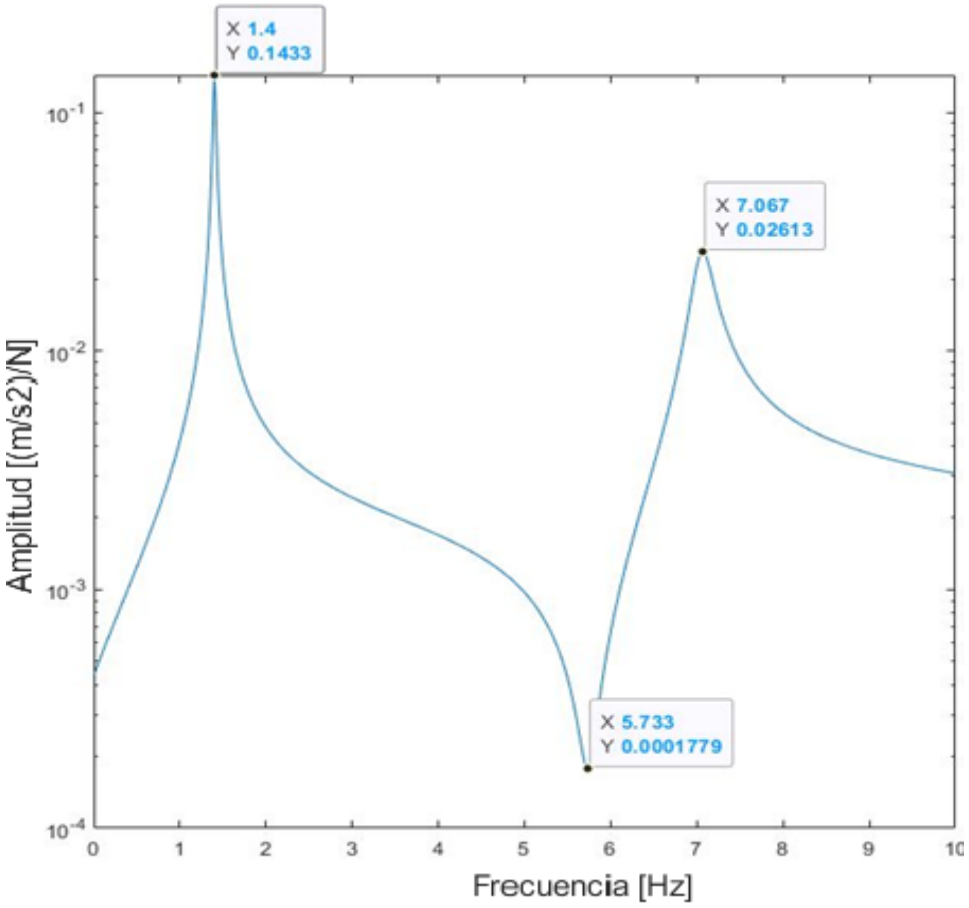

Figura 4.2FRF de la plataforma con un peso de 50kg extra en el punto 14 medida en ese mismo punto

4.2 Modelo Gris

Análogamente al capítulo anterior, el modelo Gris vuelve a ser una copia del modelo Blanco al que se le añade una masa en el nodo 14 incrementándose y así hallar cuál se parece más al modelo Blanco Manipulado por comparación de sus funciones de respuesta en frecuencia.

Para ello, al ejecutarse el código desde MATLAB®, se abre el modelo y se realizan las sucesivas iteraciones para encontrar la masa con la que se obtiene la menor diferencia de la FRF con respecto al caso envejecido. Para realizar las iteraciones se crea un bucle *for* que modifica la masa en el punto 14 con el intervalo deseado. Tras ello, y dentro del bucle, se calcula el error de la misma forma que se hizo anteriormente mediante la ecuación 10. De cada iteración se calcula su error y se compara con el error mínimo existente hasta ese momento para ver cuál es menor. Esto, al igual que en la ocasión anterior, se realiza mediante una condición *if* en la que, si en nuevo error calculado es menor que el mínimo que hay registrado, se considere el nuevo error como mínimo y también se registre la masa con la que se obtiene dicho error.

Además, hay que tener en cuenta la masa del *shaker*, que es de 40 kg y se encuentra en el punto 14 donde se mide la función de respuesta en frecuencia, por lo que habrá que sumarla

a la masa añadida en cada aumento. Dicha masa correspondiente al dispositivo de excitación podría añadirse de distintas maneras. Una opción es haciendo que el intervalo del bucle *for* empezara en 40 y cuando se obtuviera el resultado tener en cuenta que a la masa obtenida habría que restarle esos 40 kg. Sin embargo, en esta ocasión se ha decidido que el intervalo del bucle comience en 0 y haga referencia a la masa añadida extra, no a la total, y ya dentro del bucle sumar la del *shaker* antes de añadírsela definitivamente al modelo. Esto se ha realizado así en previsión del escenario del siguiente capítulo, en el que habrá que obtener la masa en un punto desconocido en el que puede estar el dispositivo o no, sumándole la masa del *shaker* solo en el caso de estar simulando el punto 14.

4.3 Resultados

Como ya se ha señalado, la primera iteración a realizar por el bucle es la de masa añadida extra de 0 kg, añadiéndole después los 40 kg correspondientes al *shaker*. Este caso viene a ser el mismo que el del modelo original en su puesta en servicio. De esta forma, si se compara dicho caso con el que se veía anteriormente con una masa extra de 50 kg se obtienen la siguiente gráfica:

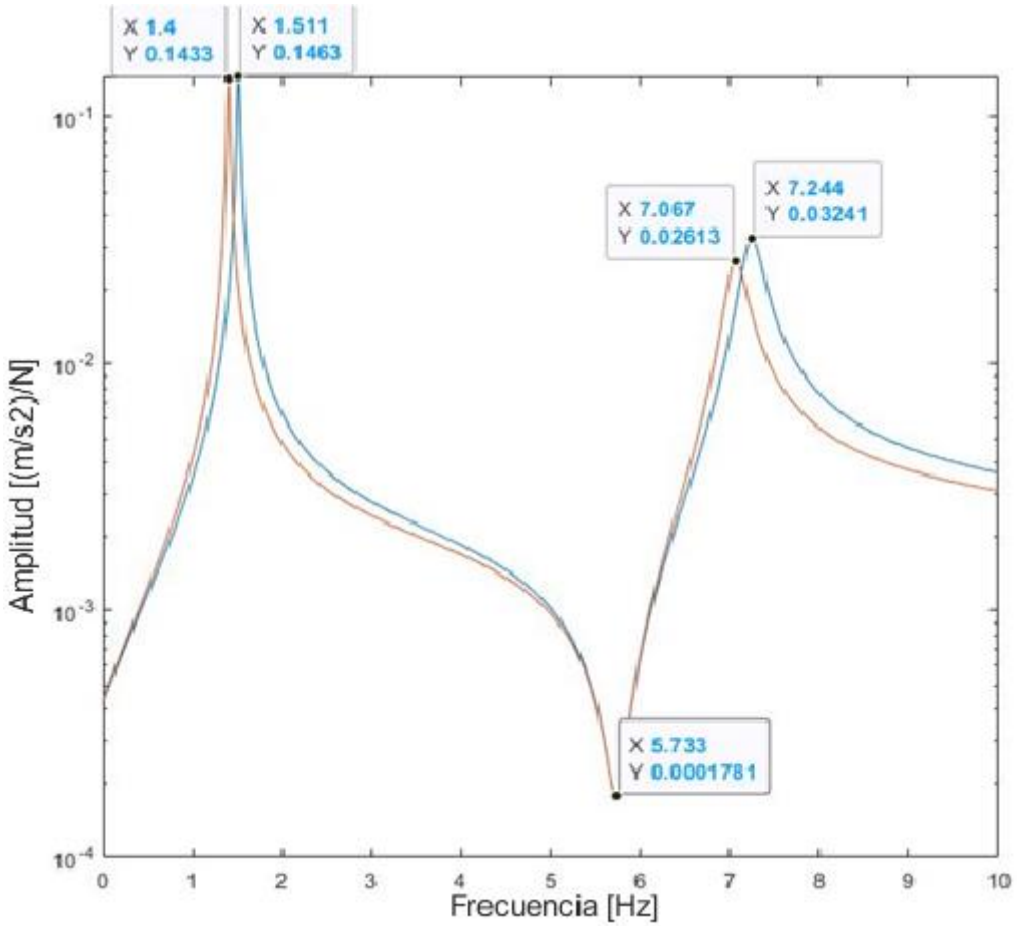

Figura 4.3 Comparación FRFs en el punto 14 para un caso con un peso extra en el punto 14 de 50 kg y otro sin peso extra

En esta comparación se observa que el pico máximo aparece a una frecuencia más baja en esta ocasión y es ligeramente inferior. El pico mínimo, por su parte, difiere muy poco con el original, sin embargo, tras el mínimo, y a diferencia de lo que se obtenía en el capítulo anterior, sí que existe una disparidad entre ambas curvas siendo muy notable en el tercer pico.

Para analizar los resultados correctamente se obtienen los errores mínimos obtenidos para diferentes ejemplos. También en este caso se hacen las simulaciones con distintos incrementos de masa comenzando con aumentos de 10 en 10 kg. Se observa que los resultados se acercan al original, con errores relativos bajos y consiguiendo resultados de masa exactos con error entre FRF nulos en aquellas masas múltiplos de 10.

Masa añadida (kg)	Masa p14 total (kg)	Incremento (kg)	Masa p14 gris obtenido (kg)	Error frf	Error relativo (%)
50,00	90,00	10	90	0	0,00%
42,00	82,00	10	80	0,1506	2,44%
33,00	73,00	10	70	0,2383	4,11%
25,00	65,00	10	70	0,4042	7,69%
21,30	61,30	10	60	0,1054	2,12%
18,00	58,00	10	60	0,1698	3,45%
13,20	53,20	10	50	0,2736	6,02%
12,50	52,50	10	50	0,2120	4,76%
10,00	50,00	10	50	0	0,00%
8,60	48,60	10	50	0,1250	2,88%

Tabla 18 Resultados del error para distintos casos de peso extra en el punto 14 con un intervalo de 10 kg

Si se baja el incremento de masa a 1, los errores relativos para los ejemplos estudiados se mantienen por debajo del 1% mientras que el error total entre ambas gráficas se ve que es mucho menor que en el caso anterior, consiguiéndose muchos más casos en los que el error es 0.

Masa añadida (kg)	Masa p14 total (kg)	Incremento (kg)	Masa p14 gris obtenido (kg)	Error frf	Error relativo (%)
50,00	90,00	1	90	0	0,00%
42,00	82,00	1	82	0	0,00%
33,00	73,00	1	73	0	0,00%
25,00	65,00	1	65	0	0,00%
21,30	61,30	1	61	0,0247	0,49%
18,00	58,00	1	58	0	0,00%
13,20	53,20	1	53	0,0176	0,38%
12,50	52,50	1	52	0,0436	0,95%
10,00	50,00	1	50	0	0,00%
8,60	48,60	1	49	0,0360	0,82%

Tabla 19 Resultados del error para distintos casos de peso extra en el punto 14 con un intervalo de 1 kg

Por último, con un incremento de 0,1 kg se consigue que el error de todos los casos estudiados sea 0. Esto es debido a que en os ejemplos que se han realizado solamente contaban con un decimal. El gran inconveniente en esta ocasión, al igual que pasaba en el capítulo anterior, es que el tiempo necesario para realizar el análisis es muy elevado.

Masa añadida (kg)	Masa p14 total (kg)	Incremento (kg)	Masa p14 gris obtenido (kg)	Error frf	Error relativo (%)
50,00	90,00	0,1	90,00	0	0,00%
42,00	82,00	0,1	82,00	0	0,00%
33,00	73,00	0,1	73,00	0	0,00%
25,00	65,00	0,1	65,00	0	0,00%
21,30	61,30	0,1	61,30	0	0,00%
18,00	58,00	0,1	58,00	0	0,00%
13,20	53,20	0,1	53,20	0	0,00%
12,50	52,50	0,1	52,50	0	0,00%
10,00	50,00	0,1	50,00	0	0,00%
8,60	48,60	0,1	48,60	0	0,00%

Tabla 20 Resultados del error para distintos casos de peso extra en el punto 14 con un intervalo de 0.1 kg

4.3.1 Resumen de los resultados obtenidos

Viendo los resultados obtenidos en este capítulo, el error mínimo se obtiene, independientemente del intervalo empleado en el bucle, para la masa más cercana a la que se ha añadido en el modelo Blanco Manipulado. Por lo tanto, en este modelo se obtienen resultados satisfactorios que permiten validar el objetivo planteado.

Capítulo 5. METODOLOGÍA PARA LOCALIZAR LA ZONA DONDE SE HA CAMBIADO LA MASA Y CUANTIFICARLA

Finalmente, el último escenario es una evolución del anterior. En este caso no solo consiste en encontrar el valor de una masa, cómo se hacía en el capítulo 4, sino también determinar el punto en el que se localiza dicha masa.

Como se ve en la Figura 5.1 el modelo calibrado de la pasarela está compuesto por 99 puntos, aunque estos no tienen una numeración continua. Los primeros 27 puntos, los cuales están resaltados en amarillo están numerados del 1 al 27, estos puntos se corresponden con un mallado inicial en el que las divisiones que se obtienen consisten en rectángulos alargados. El resto de los puntos están numerados del 82 al 153, los cuales provienen de la división de los rectángulos para conseguir dividir en superficies más regulares y así que los distintos análisis que se realizan con este modelo obtengan mejores resultados. Sin embargo, pese a hacerse las simulaciones con la estructura dividida en 99 puntos, la localización de las masas desconocidas se ha decidido limitarse a los 27 primeros puntos, los cuales están repartidos en 9 secciones, entre las que se incluyen ambos extremos y el centro de la estructura. Esto se hace para reducir el tiempo necesario, ya que para cada simulación completa del modelo es necesario simular cada punto varias veces con diferentes valores de masa.

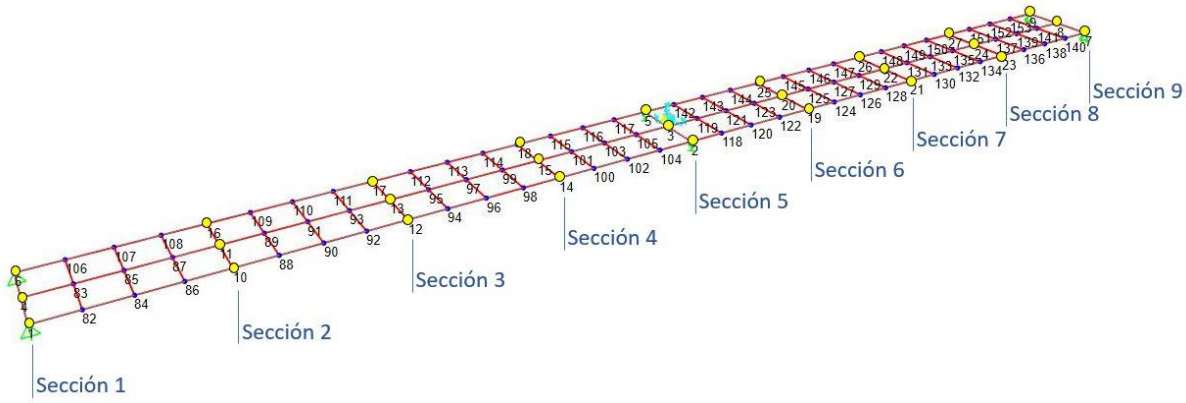

Figura 5.1 Localización de los 27 puntos elegidos y sus respectivas secciones

5.1 Modelo Blanco Manipulado

El modelo Blanco Manipulado en este caso es análogo al del Capítulo 4 con la salvedad de que no se remite a ubicar la masa extra en el punto 14, sino que puede ir en cualquiera de los 27 primeros puntos. Además, hay que tener en cuenta de que en dicho punto ha de incluirse también la masa del *shaker* independientemente del punto en el que se encuentre la masa extra.

5.2 Modelo Gris

Para llevar a cabo el modelo Gris en este caso se hace uso de dos bucles *for*. Primero un bucle *for* que recorre los 27 puntos y dentro de él, otro bucle *for* que aumenta la masa desde 0 hasta la cifra que se considere necesaria con la precisión que se haya requerido.

También se han añadido condiciones para que en cada simulación no solo se modifique la masa del punto en cuestión, sino también que se tenga en cuenta la masa del *shaker* en el

punto 14, y en caso de estar estudiando el aumento de masa en dicho punto, que se tengan ambas en cuenta, como se muestra en la Figura 5.2.

```
%bucle masa
for m = 0:10:50 %Indicar precisión aquí / comienza en masa 0 kg

    m1 = m;

    %%%%%%%%%%%%%%%%%%
    % 2) Modificar masa

     if pos == 14 %en el caso de ser pto 14 hay que añadir masa shaker al total

        m1 = m1 + m_shk;

        point = num2str(pos); % Punto en que se aplica la carga
        mass = [m1, m1, m1, 0, 0, 0];
        elemType = s2k.Objects;
        localCSys = false;
        replace = true;
        s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

      else %Si es en otro punto hay que aplicar la del shaker en 14 y la otra en el punto elegido
        point = num2str(pos); % Punto en que se aplica la carga
        mass = [m1, m1, m1, 0, 0, 0];
        elemType = s2k.Objects;
        localCSys = false;
        replace = true;
        s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

        point = '14'; % Punto en que se aplica el shaker
        mass = [m_shk, m_shk, m_shk, 0, 0, 0];
        elemType = s2k.Objects;
        localCSys = false;
        replace = true;
        s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);
    end
```

Figura 5.2 Detalle de las condiciones en el script de MATLAB

El hecho de añadir también la masa del *shaker* en el punto 14 en cada ocasión, a pesar de ser siempre la misma, se debe a que previo a añadir las masas se eliminan todas las masas existentes que pueda haber de simulaciones anteriores para así no obtener resultados erróneos. Además, aunque el modelo de la pasarela físicamente es simétrico, dicha masa en el punto 14 producida con el *shaker* hace que a la hora de simularlo deje de serlo tomando la sección de los apoyos móviles como eje de simetría, por lo que no existen problemas de simetría entre los resultados de añadir la masa en un punto o en el mismo punto del lado opuesto.

También es importante tener en cuenta que aquellas masas que se encuentran en los apoyos (puntos 1, 6, 7 y 9) dan resultados que pueden llevar a error ya que la fuerza que generan es absorbida por los apoyos, por lo que no se ve reflejado en una modificación de las FRF.

A la hora de comparar los resultados con respecto al de Blanco Manipulado vuelve a utilizarse la ecuación 11 al igual que en los escenarios anteriores. Igualmente vuelve a compararse el error de cada iteración con el error mínimo hasta ese momento y en caso de ser menor es este último el que se convierte en el error mínimo quedando registrado para qué masa y en qué punto.

5.3 Resultados

Se han realizado simulaciones de distintas masas en puntos diferentes de la estructura para ver los resultados que se obtienen. Esta vez se han realizado únicamente con una precisión de 10 kg ya que al tener que realizar varias iteraciones en los 27 puntos, reducir la precisión implica que la duración necesaria para realizar el análisis completo sea extremadamente larga. Además, ya se ha visto en el capítulo anterior cómo el ir reduciendo la precisión tiene un funcionamiento correcto a la hora de buscar la masa exacta.

A continuación, se presentan resultados obtenidos para ejemplos con una masa aplicada de 18 kg en distintos puntos aleatorios que vienen señalados en la Figura 5.3.

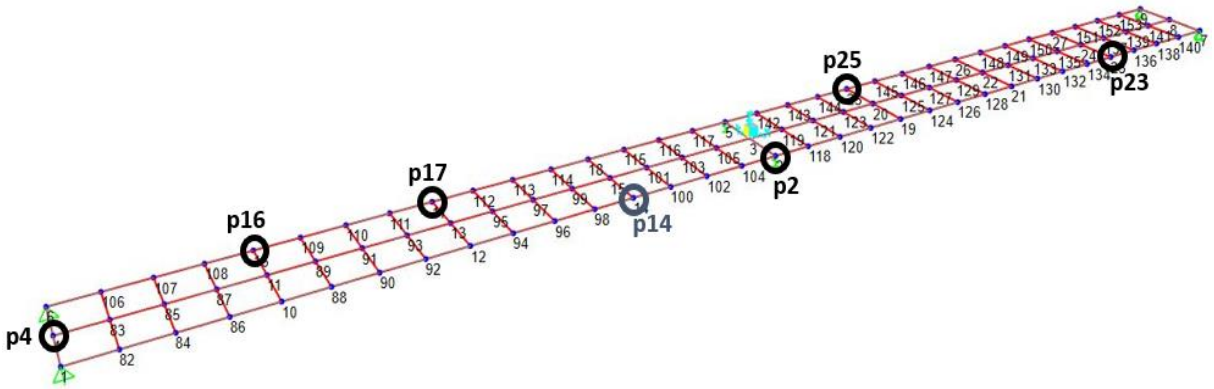

Figura 5.3 Localización de los puntos elegidos para los distintos ejemplos propuestos

En todos los casos estudiados en la Tabla 21, los errores más pequeños se presentan en los puntos de la sección en la que está aplicada la masa, pero no siempre el error mínimo es en el punto exacto, sino en otro de la misma sección, aunque por diferencias mínimas. Sin embargo, sí que se observa que en el resto de las secciones los errores en sus puntos son notablemente mayores, quedando así descartados.

En los resultados se ha resaltado el valor mínimo obtenido, en verde en el caso de que coincida con el punto donde está la masa aplicada y en rojo si dicho valor del error mínimo aparece en otro punto.

Punto 2 m=18kg	**Puntos de la sección 5**		
Punto	**2**	3	5
Masa (kg)	**20**	20	20
Error FRF	**0,146319815**	0,14768799	0,14632658
Punto 4 m=18kg	**Puntos de la sección 1**		
Punto	1	**4**	6
Masa (kg)	Cualquiera	**20**	Cualquiera
Error FRF	6,59E-05	**7,34E-06**	6,59E-05
Punto 16 m=18kg	**Puntos de la sección 2**		
Punto	**10**	11	16
Masa (kg)	**20**	20	20
Error FRF	**0,07270018**	0,07230540	0,07270020

Punto 17 m=18kg	Puntos de la sección 3		
Punto	12	13	17
Masa (kg)	20	20	20
Error FRF	0,15909330	0,15840949	0,15909899
Punto 23 m=18kg	Puntos de la sección 8		
Punto	23	24	27
Masa (kg)	20	20	20
Error FRF	0,08029190	0,07987299	0,08029190
Punto 25 m=18kg	Puntos de la sección 6		
Punto	19	20	25
Masa (kg)	20	20	20
Error FRF	0,18774536	0,18723630	0,18774540

Tabla 21 Resultados del error para una masa de 18 kg aplicado en distintas posiciones

Como puede observarse en estos seis ejemplos distintos, el error mínimo es muy similar en todos los puntos de la sección, aunque la masa solo se aplique en uno de ellos, pero, como se ha comentado, el error mínimo no siempre aparece en el punto aplicado, sino en otro de la misma sección, aunque sea por una diferencia mínima.

También se observa con el primer ejemplo en el que se aplican los 18 kg en el punto 4 que, en los apoyos, que son los otros nodos de la sección 1, se absorbe la fuerza generada por la masa por lo que el error en ellos es siempre el mismo independientemente de la masa aplicada.

Además, también se ha querido comprobar cuál es el resultado en los puntos de la sección simétrica para ver si podrían existir similitudes por la simetría. Por ello se muestran en la Tabla 22 la diferencia de los errores de las secciones anteriores frente a los errores de la sección opuesta de cada una a excepción de la sección 5, correspondiente al punto 2, que se encuentra en el eje de simetría.

Punto 4 m=18kg	Puntos de la sección 1			Puntos de la sección 9		
Punto	1	4	6	7	8	9
Masa (kg)	Cualquiera	20	Cualquiera	Cualquiera	20	Cualquiera
Error FRF	6,59E-05	7,34E-06	6,59E-05	6,59E-05	2,27E-05	6,59E-05
Punto 16 m=18kg	Puntos de la sección 2			Puntos de la sección 8		
Punto	10	11	16	23	24	27
Masa (kg)	20	20	20	20	20	20
Error FRF	0,07270018	0,07230540	0,07270020	0,25331544	0,25319163	0,25331543
Punto 17 m=18kg	Puntos de la sección 3			Puntos de la sección 7		
Punto	12	13	17	21	22	26
Masa (kg)	20	20	20	20	20	20
Error FRF	0,15909330	0,15840949	0,15909899	0,57446990	0,57437824	0,57446989
Punto 23 m=18kg	Puntos de la sección 8			Puntos de la sección 2		

Punto	23	24	27	10	11	16
Masa (kg)	20	20	20	20	20	20
Error FRF	0,08029190	0,07987299	0,08029190	0,27670037	0,27650537	0,27670045
Punto 25 m=18kg	Puntos de la sección 6			Puntos de la sección 7		
Punto	19	20	25	14	15	18
Masa (kg)	20	20	20	20	20	20
Error FRF	0,18774536	0,18723630	0,1877454	0,56113915	0,56606175	0,56690666

Tabla 22 Resultados del error para una masa de 18 kg aplicado en distintas posiciones

También se ha probado con masas diferentes, las cuales se han escogido de forma aleatoria, en los mismos puntos de los ejemplos anteriores, obteniéndose resultados semejantes que se muestran en la Tabla 23.

Punto 2 m=28kg	Puntos de la sección 5		
Punto	2	3	5
Masa (kg)	30	30	30
Error FRF	0,142831491	0,14465022	0,14284161
Punto 4 m=19kg	Puntos de la sección 1		
Punto	1	4	6
Masa (kg)	Cualquiera	20	Cualquiera
Error FRF	3,30E-05	3,67E-06	3,30E-05
Punto 16 m=13kg	Puntos de la sección 2		
Punto	10	11	16
Masa (kg)	10	10	10
Error FRF	0,10936233	0,10946499	0,10936232
Punto 17 m=8kg	Puntos de la sección 3		
Punto	12	13	17
Masa (kg)	10	10	10
Error FRF	0,17160656	0,17141593	0,17160885
Punto 23 m=10kg	Puntos de la sección 8		
Punto	23	24	27
Masa (kg)	10	10	10
Error FRF	0	1,06E-04	2,12E-10
Punto 25 m=20kg	Puntos de la sección 6		
Punto	19	20	25
Masa (kg)	20	20	20
Error FRF	9,60E-08	5,09E-04	0

Tabla 23 Resultados del error para distintas masas aplicadas en distintas posiciones

Si bien es este caso se consiguen los errores mínimos en los puntos exactos en más ocasiones fundamentalmente porque en dos de los ejemplos la masa elegida puede alcanzarse con exactitud con la precisión definida.

Aun así, debido a que en varios de los casos estudiados el punto mínimo no ha coincidido con el exacto, sino con otro de su sección, se ha decidido comprobar si esto se podría resolver analizando las funciones de respuesta en frecuencia (FRF) en varios puntos, en vez de solo el del punto 14.

Para ello se ha elegido el caso del punto 25 con 18 kg de masa donde ya se ha visto que el error mínimo no coincide con el punto exacto. En el Anexo III pueden verse los errores que se obtienen para este caso obteniendo las FRFs en distintos puntos. A continuación, en la Tabla 24 se han extraído los resultados de los puntos de la sección 6 con los errores que se obtienen midiendo las funciones de respuesta en frecuencia en 11 puntos distintos, 6 en el lateral dónde se encuentra el *shaker*, 2 en el centro y 2 en el lateral opuesto. Estas últimas cuatro FRFs pertenecen a puntos de las secciones donde se encuentra el *shaker* (sección 4), y también en el que está el punto en el que se aplica la masa (sección 6) indicadas en la Figura 5.4.

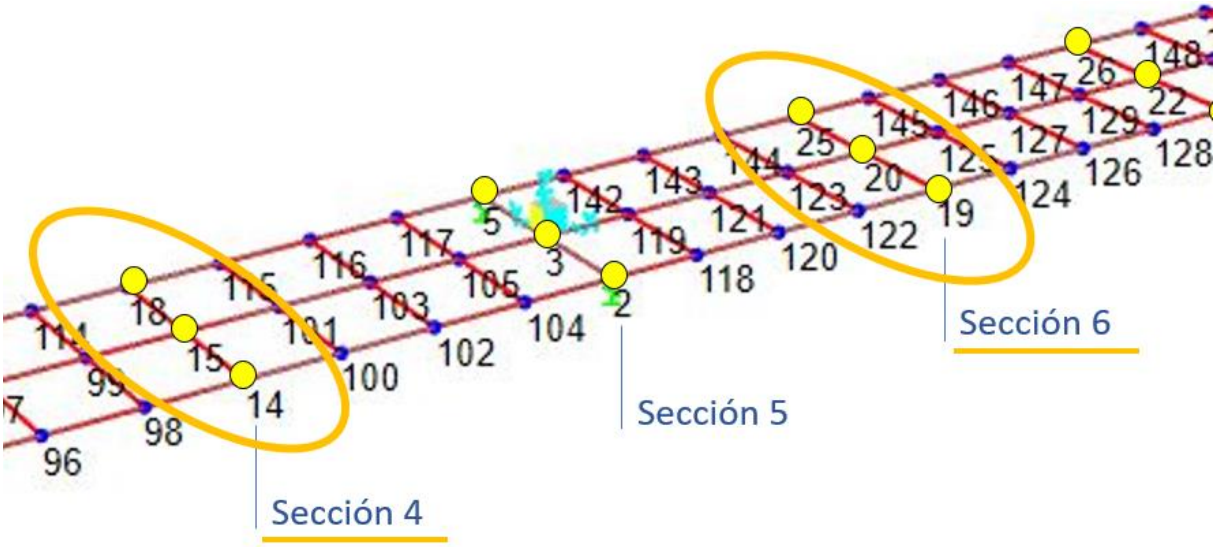

Figura 5.4 Localización de los puntos de las secciones 4 y 6

Punto 25 m=18kg		Error para FRFs en distintos puntos para una masa de 18 kg en el punto 25										
		Lateral 1							Centro		Lateral 2	
Pto.	M (kg)	L/8 Error FRF10	2L/8 Error FRF12	3L/8 Error FRF14	4L/8 Error FRF2	5L/8 Error FRF19	6L/8 Error FRF21	7L/8 Error FRF23	3L/8 Error FRF15	5L/8 Error FRF20	3L/8 Error FRF18	5L/8 Error FRF25
19	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,4491	0,7228	0,7347	0,5683	0,7303	0,7154	0,4451	0,7342	0,7298	0,7342	0,7293
	20	0,1134	0,1832	0,1877	0,1465	**0,1849**	0,1805	0,1119	0,1876	0,1856	0,1875	0,1864
20	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,4492	0,7230	0,7351	0,5839	0,7299	0,7156	0,4452	0,7344	0,7301	0,7343	0,7295
	20	**0,1130**	**0,1826**	**0,1872**	**0,1463**	0,1850	**0,1798**	**0,1114**	**0,1871**	**0,1848**	**0,1870**	0,1858
25	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,4491	0,7228	0,7350	0,5683	0,7298	0,7154	0,4451	0,7342	0,7298	0,7342	0,7298
	20	0,1134	0,1832	0,1877	0,1465	0,1856	0,1805	0,1119	0,1876	0,1856	0,1875	**0,1856**

Tabla 24 Resultados del error para el caso de 18 kg en el punto 25 medido en distintos puntos

Viendo los resultados puede observarse que el hecho de medir las FRFs en más puntos no hace que se obtenga un resultado mejor, ya que, de los 11 puntos desde los que se comprobado las funciones de respuesta en frecuencia, únicamente coinciden el punto con error mínimo y el punto que se busca, señalado en verde en la tabla, en el caso en el que la FRF está medida en el punto donde está la masa aplicada (FRF del punto 25). También se observa que, aunque mayoritariamente en este caso el error mínimo se encuentra en el punto central de la sección, existen dos casos en los que no. Sin embargo, lo que si queda claro es que independientemente de la FRF que se estudie, para todos los casos el error mínimo se encuentra en la misma sección que es la que se aplica la masa.

5.3.1 Resumen de los resultados obtenidos

Tras el análisis de los resultados obtenidos, el modelo es capaz de determinar la masa más cercana a la que se busca y los errores más bajos se dan para los puntos de la sección donde se localiza dicha masa, sin embargo, no es capaz de determinar con exactitud el punto correcto de la sección donde está aplicada. Por lo que el modelo sí que puede servir al menos para determinar la sección en la que se localiza la nueva masa y determinar cuánto es.

Capítulo 6. CONCLUSIONES, REPERCUSIONES Y LÍNEAS FUTURAS

6.1 Conclusiones

Tras la conclusión del presento documento y después de haber realizado todos los ensayos necesarios y habiendo alcanzado los objetivos al inicio de este Trabajo de Fin de Máster, puede llegarse a las siguientes conclusiones.

En primer lugar, se ha visto que es viable la realización de una API mediante MATLAB® con la que analizar las distintas modificaciones planteadas a una estructura, en este caso un modelo digital, y realizar su comparación con su gemelo digital en SAP2000®.

Para los experimentos planteados en el capítulo 3, puede concluirse que el modelo realizado es perfectamente capaz de determinar la modificación en la rigidez cuando es común en ambos apoyos elásticos. También consigue cuantificar el deterioro cuando la rigidez disminuye únicamente en uno de los apoyos elásticos, aunque no logra determinar con exactitud cuál de los dos apoyos es el dañado debido a la simetría. Además, se ha visto que, debido a la simetría, no es capaz de determinar correctamente la rigidez de ambos apoyos elásticos cuando ambos se ven dañados de forma desigual.

En el caso de determinar la masa en un punto conocido, puede concluirse que el modelo consigue dar el resultado más cercano al que se busca dependiendo de la precisión que se le dé, dando el resultado exacto cuando la presión asignada lo permite.

Otra conclusión a la que se puede llegar es que, ante una masa desconocida en un punto también desconocido, el modelo propuesto es capaz de cuantificar la masa, así como determinar la sección en la que se encuentra dicha masa, sin embargo, no consigue precisar con exactitud el punto concreto de la sección en la que se encuentra.

6.2 Repercusiones económicas

Para estimar la repercusión económica que tiene este trabajo se van a tener en cuentan distintos tipos de gasto:

- Costes directos: que afectan de manera directa al valor final del producto y se van a dividir en este caso de la siguiente forma:
  - Mano de obra directa
  - Coste de amortización del software
  - Coste de amortización del hardware
- Costes indirectos: que no afectan directamente al producto.

6.2.1 Mano de obra directa

Lo primero que hay que tener en cuenta es el tiempo que se ha invertido en su realización y su coste. Para la realización de este TFM se han invertido aproximadamente 300 horas entre estudio previo (10%), ejecución (60%) y redacción del mismo (30%) que se corresponden con las horas de dedicación necesarias para 12 ECTS.

Para el cálculo del coste es necesario obtener el número de horas que se trabajan al año teniendo en cuenta vacaciones, fines de semana, festivos y días para asuntos personales, enfermedad etc.

Asunto	Días
Año	365
-Fines de semana (52 al año)	-104
-Vacaciones	-20
-Festivos	-12
-Asuntos personales o enfermedad	-15
TOTAL	**214**

Tabla 25 Días efectivos en un año

Se llega a la conclusión de que al año hay unas 214 jornadas laborales que a 8 horas diarias de trabajo suponen 1712 horas de trabajo al año. Considerando que un ingeniero en formación puede percibir un sueldo de unos 22000€ al año trabajando se obtiene que por cada hora de trabajo calculada se perciben aproximadamente 12,85€. Por lo que el coste de mano de obra directa es:

$$Mano\ de\ obra\ directa = 300\ horas \cdot 12{,}85€/\mathrm{h} = 3855€$$

6.2.2 Software

Si bien todos los programas de software se encontraban disponibles en la Escuela de Ingenierías Industriales y por lo tanto no ha hecho falta comprar de forma exclusiva para la realización de este TFM, si que se han tenido en cuenta a la hora de calcular los costes que supondrían para su realización. En el caso de MATLAB® una licencia anual tiene un coste de 800€, pero también existen licencias educacionales como la que se ha utilizado durante el desarrollo de este TFM y que son perfectamente válidas por 250€. Al consistir en una licencia con una duración de 1 año la amortización se considera de solo un año.

Por su parte, SAP2000® tiene distintos tipos de licencia, además de la educacional, aunque de esta no se ha conseguido el precio. Del resto, la licencia *Basic* no incluye el análisis de estado estacionario (*steady-state*) por lo que habría que optar por lo menos por la licencia *Plus*, la cual tiene un precio de $5000, que al cambio son alrededor de 5000€ en el momento en el que se realizó este TFM. En este caso la licencia es de carácter indefinido al adquirirla pese a que se renueva anualmente. Al existir nuevas versiones con mejoras cada poco tiempo se va a considerar una amortización lineal a 4 años en este caso.

Concepto	Coste de adquisición (€)	Hora de vida útil (h)	Coste por hora (€/h)
Licencia MATLAB®	250	1712	0,15
Licencia SAP2000®	5000	6848	0,73

Tabla 26 Costes de las licencias de software

Considerando que el uso de ambos programas se va a realizar durante un 60% del tiempo empleado para este TFM:

$$Coste\ amortización\ software = (0{,}15€ + 0{,}73€) \cdot 180\ \mathrm{h} = 158{,}40€$$

6.2.3 Hardware

El siguiente cálculo es el del coste de amortización del equipo necesario para realizar el trabajo. Para ello es necesario un ordenador portátil y para ello se ha considerado un HP Envy

13-ba1004ns de 877,49€. Para el equipo informático se considera que se amortiza linealmente en 4 años.

Concepto	Coste de adquisición (€)	Hora de vida útil (h)	Coste por hora (€/h)
Ordenador portátil	877,49	6848	0,13

Tabla 27 Costes del hardware

Considerando que se va a utilizar durante las 300 horas que lleva la realización del TFM el coste de amortización es:

$$Coste\ amortización\ hardware = 0{,}13€/\mathrm{h} \cdot 300\ \mathrm{h} = 39€$$

6.2.4 Costes indirectos

El coste indirecto principal que hay que calcular es el del consumo de electricidad por parte del equipo informático. La potencia consumida por el ordenador se estima en unos 200 W. Teniendo en cuenta que se utiliza durante las 300 horas que dura el TFM el consumo será aproximadamente de unos 60 kWh. Teniendo en cuenta un precio medio de 0,29 €/kWh:

Concepto	Consumo eléctrico (kWh)	Precio electricidad (€/kWh)
Coste electricidad	60	0,29

Tabla 28 Costes indirectos

$$Coste\ electricidad = 0{,}29€/\mathrm{kWh} \cdot 60\ \mathrm{kWh} = 17{,}40€$$

6.2.5 Costes totales

Finalmente, se recogen todos los costes obtenidos en los apartados anteriores en la siguiente tabla:

Apartado	Coste (€)
Costes Directos	
Coste de mano de obra directa	3855 €
Coste de amortización del software	158,40 €
Coste de amortización del hardware	39 €
Costes Indirectos	
Coste electricidad	17,40 €
TOTAL	**4069,64** €

Tabla 29 Costes totales

Viendo la Tabla 29 se puede concluir que el coste total estimado que conlleva la realización de este TFM es de 4069,64 €.

6.3 Repercusiones ambientales

Para la elaboración de este Trabajo de Fin de Máster, hay que señalar que no ha sido necesario emplear ningún material peligroso ni contaminante que pudiera ser tenido en consideración. En cambio, sí que podrían existir impactos ambientales en un futuro al implementarse este método en modelos físicos por la necesidad de los distintos dispositivos y sensores para las mediciones necesarias.

6.4 Consideraciones adicionales

Cabe destacar que la realización de este Trabajo de Fin de Máster ha permitido no solo poner en práctica conocimientos adquiridos durante el Grado y el Máster, sino ampliarlos y también poder adquirir otros nuevos. Prueba de ello es que a las competencias ya adquiridas en el uso de los programas de software MATLAB® y SAP2000® utilizados en distintas asignaturas se le añaden nuevas como son el desarrollo de APIs entre ambas o el uso de patrones de carga nuevos que no se habían estudiado durante los estudios. También la profundización en conceptos como *Structural Health Monitoring* o las funciones de respuesta en frecuencia (FRF).

Además, este trabajo supone una iniciación a la investigación, lo cual puede ser muy útil en cuanto a la visión y capacidades que aporta de cara al futuro y al desarrollo personal en la vida laboral. También con su realización se ha conseguido mejorar la capacidad de organización planificación, así como el trabajo de forma autónoma.

6.5 Líneas futuras

A continuación, se van a exponer las distintas líneas de desarrollo que se pueden seguir para continuar con el proyecto.

En primer lugar, parece importante poder reducir los tiempos necesarios para realizar el análisis de la estructura. Para ello, como línea futura se propone, además de mejorar los dispositivos con los que realizar los análisis y cálculos, también implementar una optimización en los códigos realizados con MATLAB® para no tener que realizar todo el rango de valores posible, lo cual hace que el análisis se alargue demasiado. Como ejemplo, en el caso de rigidez común en ambos apoyos elásticos, esto puede llevarse a cabo mediante un método de bisección. Comenzando con el bucle inicial con un intervalo de 1000 N/m se obtienen los valores de las dos rigideces para los que el error es mínimo, tras ello, se realiza un bucle de intervalo 100 N/m entre los dos valores extraídos. Al acabar el bucle, nuevamente se registran las rigideces de los dos valores del error mínimo y se realiza un nuevo bucle de intervalo 10 N/m entre dichos valores y así sucesivamente hasta 0,1 N/m o hasta obtener la solución exacta.

En segundo lugar, tratándose este TFM de un inicio de investigación para la búsqueda de un método con el que determinar el deterioro de estructuras en el que se ha realizado todo mediante software, una línea futura clara para continuar el trabajo sería su implementación en el modelo físico existente en el laboratorio. Al ya existir dicha estructura, simplemente haría falta conseguir cambiar las rigideces y las masas dependiendo el caso para simular un deterioro. Tan solo serían necesarios muelles de distintas rigideces para modificar los apoyos elásticos del primer escenario y distintos pesos que colocar en puntos de la estructura. Por otro lado, para hacer los ensayos serían necesarios acelerómetros, un *shaker* como dispositivo de excitación y el software necesario para obtener todos los datos.

Una vez conseguidos todos los elementos habrá que realizar simulaciones sobre la estructura física y ver si el modelo es capaz de determinar el deterioro correctamente al igual que se ha realizado en este trabajo. Para ello habrá que tener en cuenta los posibles ruidos que puedan interferir al tratarse de un modelo real y que puedan hacer que los resultados no sean los correctos.

Además, y siguiendo el cauce natural de la investigación sería la implementación del método en estructuras reales en uso y estudiar a lo largo del tiempo si existen cambios debido al deterioro y comprobar si funciona correctamente.

Por último, otra línea futura que se plantea es la de investigar otras posibles modificaciones como pueden ser el cambio de apoyos y de las condiciones de contorno, que en la realidad pueden deberse a asentamientos o descalces, aflojamiento o rotura de los tornillos de la estructura, holguras, envejecimiento, en el caso de la estructura del laboratorio este será de las propiedades generales de la madera, o humedades entre otros.

REFERENCIAS

[1] J. Monjo, "Durabilidad vs Vulnerabilidad", *Informes de la Construcción*, vol. 59, 507, 43-58, 2007. [Online] Disponible en: https://digital.csic.es/bitstream/10261/23071/1/606.pdf

[2] A. B. Menéndez, G. Arias y M. L. Ramírez, Guía para la realización de inspecciones principales de obras de paso en la Red de Carreteras del Estado, *Dirección General de Carreteras - Ministerio de Fomento*, Madrid, 2012. [Online] Disponible en: https://www.mitma.gob.es/recursos_mfom/0870250.pdf

[3] Recomendaciones para la realización de pruebas de carga de recepción en puentes de carretera, *Dirección General de Carreteras - Ministerio de Fomento*, Madrid, 1999. [Online] Disponible en: https://www.mitma.es/recursos_mfom/0850100.pdf

[4] D. Adhikari, A. Prakash, S. Yadav y S. Kaloni, "Structural Health Monitoring", 2019. [Online] Disponible en: https://www.researchgate.net/publication/341724905_Structural_Health_Monitoring

[5] D. Balageas, "Introduction to Structural Health Monitoring", en Structural Health Monitoring, D. Balageas, C. Fritzen y A. Güemes, iSTE, 2006, pp13-43. [Online] Disponible en: http://www.iste.co.uk/data/doc_xqjujdlhnfls.pdf

[6] Y. Yang, Y. Zhang y X. Tan, Review on Vibration-Based Structural Health Monitoring Techniques and Technical Codes, *Symmetry*, 2021, 13, 1998. [Online] Disponible en: https://www.mdpi.com/2073-8994/13/11/1998

[7] C. de la Fuente, Guía práctica para la publicación de Datos Abiertos usando APIs, *Ministerio de Asuntos Económicos y Transformación Digital*, junio 2020. [Online] Disponible en: https://datos.gob.es/sites/default/files/doc/file/guia_publicacion_apis.pdf

[8] CSI America (2016) CSI Analysis Reference Manual for SAP2000®, ETABS®, SAFE® and CSiBridge®. [Online] Disponible en: https://docs.csiamerica.com/manuals/sap2000/CSiRefer.pdf

[9] D. J. Ewins, *Modal testing: Theory, practice and application*. 2ª Ed., Baldock, Inglaterra: Research Studies Press, 2000.

[10] B. Balachandran, E. B. Magrab, *Vibrations.* 2ª Ed., Toronto, Canada: Cengage Learning, 2009.

ANEXO I: INSTRUCCIONES PARA LA CREACIÓN DE UNA API MATLAB® + SAP2000®

Para realizar la API de MATLAB® con SAP2000® se ha que saber cómo programar el código de MATLAB® correctamente. A continuación, se mostrarán los elementos necesarios para la realización de este trabajo. Las funciones necesarias para programar la API de las que se hablarán en este anexo pueden encontrarse en un archivo que se llama 'CSI_OAPI_Documentation.chm'.

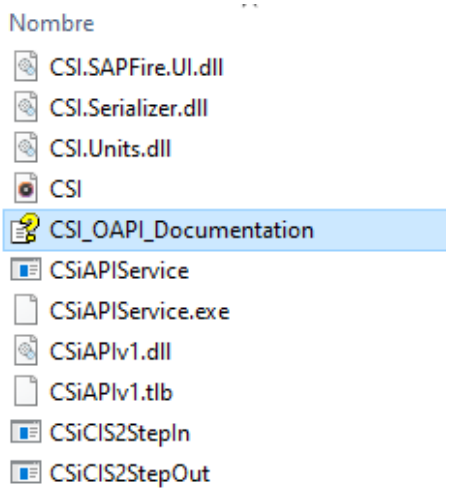

Figura Anexo1 1 Localización del archivo

Este archivo de ayuda se encuentra dentro de una carpeta la cual se obtiene al instalar el programa SAP2000®. Por defecto suele localizarse en Archivos de programa > *Computers and Structures* > *SAP2000 22*. Esta última puede diferir dependiendo de la versión. Dentro del archivo se pueden encontrar todas las funciones dentro del apartado *CSi OAPI Functions* y también ejemplos de código útiles con distintos lenguajes de programación.

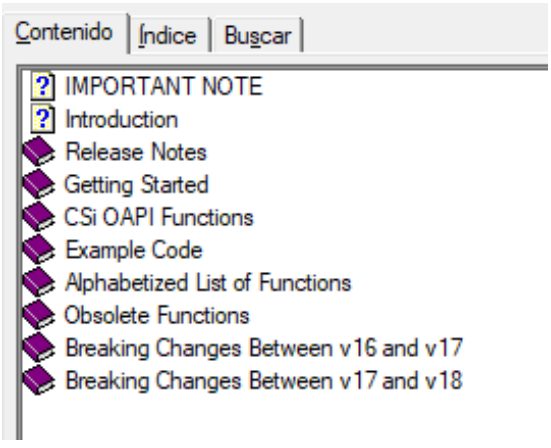

Figura Anexo1 2 Contenido del archivo

Para este trabajo se ha utilizado por un lado una función en MATLAB® para inicialización y por otro, los distintos modelos, por lo que para la explicación en este anexo se ha mantenido dicha división.

Función de inicialización

MATLAB® permite crear funciones, que son archivos que aceptan unos argumentos de entrada y devuelven unos argumentos de salida. Para este trabajo se ha hecho uso de una función de inicialización llamada *initSAPv22*.

```
function [ok, s2k] = initSAPv22(show, APIPATH)
% Función de inicialización al trabajo con SAP2000
% ENTRADA:
%   show: true/false para ver o no la ventana de SAP2K
%   APIPATH: ruta completa a un fichero de SAP2000. Si no existe, se crea.
```

Figura Anexo1 3 Declaración de la función

Los argumentos de entrada son *show*, el cual si entra como *true* mostrará la ventana de SAP2000® y si es *false* trabajará en un segundo plano, y por otro lado APIPATH que es la ruta al archivo de SAP2000®. Por otra parte, los argumentos de salida serán *ok*, que en caso de estar todo bien valdrá 0 y si existe algún problema su valor será 1, y *s2k*.

Tras ello comienza realmente la inicialización, estableciendo las banderas correspondientes a si existe (*true*) o es necesaria una instancia nueva (*false*) y la de especificar si abrir la última versión del software que esté disponible (*true*) o establecer la dirección manualmente (*false*). Después es necesaria la dirección completa donde se encuentra el archivo .dll para la API la cual se especifica en el script principal y se obtiene al nombrar la función.

```
% INICIALIZACIÓN
% Set the following flag to true to attach to an existing instance of the
% program otherwise a new instance of the program will be started.
AttachToInstance = false; % true;

% Set the following flag to true to manually specify the path to SAP2000.exe
% this allows for a connection to a version of SAP2000 other than the latest
% installation otherwise the latest installed version of SAP2000 will be launched.
SpecifyPath = false; % true; %

% If the above flag is set to true, specify the path to SAP2000 below
% ProgramPath = 'C:\Program Files\Computers and Structures\SAP2000 20\SAP2000.exe';

% Full path to API dll. Set it to the installation folder. Examples:
% APIDLLPath = 'D:\Computers and Structures\SAP2000 22\SAP2000v1.dll';
% APIDLLPath = 'C:\Program Files\Computers and Structures\SAP2000 22\SAP2000v1.dll';
APIDLLPath = APIPATH;
```

Figura Anexo1 4 Inicialización de la función

El siguiente paso es el de crear el objeto de ayuda de la API, y posteriormente, se utilizan condicionantes *if/elseif* para especificar la dirección y/o abrir una nueva instancia del programa según lo especificado en las banderas anteriores. Además, se ordena que se ejecute la aplicación de SAP2000®. Dependiendo de lo especificado en la entrada respecto a *show* se mostrará o no la ventana del programa.

```
% Create API helper object
a = NET.addAssembly(APIDLLPath);
helper = SAP2000v1.Helper;
helper = NET.explicitCast(helper,'SAP2000v1.cHelper');

if AttachToInstance
    % attach to a running instance of Sap2000
    SapObject = helper.GetObject('CSI.SAP2000.API.SapObject');
    SapObject = NET.explicitCast(SapObject,'SAP2000v1.cOAPI');
else
    if SpecifyPath
        % create an instance of the SapObject from the specified path
        SapObject = helper.CreateObject(ProgramPath);
    else
        % create an instance of the SapObject from the latest installed SAP2000
        SapObject = helper.CreateObjectProgID('CSI.SAP2000.API.SapObject');
    end
    SapObject = NET.explicitCast(SapObject,'SAP2000v1.cOAPI');

    % start Sap2000 application
    SapObject.ApplicationStart;
end

if(~show)
    SapObject.Hide;
end
```

Figura Anexo1 6 Creación del objeto API helper

Una vez iniciado el programa se crea un nuevo objeto y se inicializa un modelo. Si ocurriera algún error a la hora de inicializar el modelo, el programa asignará un valor 1 a la variable *ok* y un 0 a la variable *s2k* las cuales serán retornadas como salida al script general significando que ha existido un fallo. En cambio, si no se da la condición que implique un fallo, no habrá ningún problema.

```
% Create SapModel object
SapModel = NET.explicitCast(SapObject.SapModel,'SAP2000v1.cSapModel');

s2k.SapObject = SapObject;
s2k.SapModel = SapModel;

% Initialize model
ret = SapModel.InitializeNewModel;
if ret
    ok = 1; %% FALLO: no se ha podido inicializar.
    s2k = 0;
    return
end
```

Figura Anexo1 5 Creacióne inicialización del modelo de SAP

A partir de aquí se definen los diferentes grupos (*menu*) de propiedades, objetos, casos de carga etc. que suelen ser necesarios con frecuencia a la hora de programar un modelo. Definiéndolos en la función de inicialización permite que no sea necesario definirlos cada vez que se utilizan en el script. La forma para realizarlo es igualando a una variable el término

Net.explicitCast(_____._____, 'SAP2000v1.c_____'), siendo los huecos el término correspondiente en cada acción. Posteriormente, esa variable se iguala a otra de la forma *s2k.*_____ que será la salida de información al script principal. De esta forma, como se ha señalado, en el script principal solo será necesario nombrar la función correspondiente a la acción que se quiera realizar con los datos o entradas que sean requeridas.

El primer apartado que aparece es el de *File menu* que es utilizado para crear un nuevo modelo en blanco de SAP2000®. Seguido de él se define los términos utilizados para definir un grupo.

```
% File menu
File = NET.explicitCast(SapModel.File,'SAP2000v1.cFile');
s2k.File = File;

% Definitions menu
GroupDef = NET.explicitCast(SapModel.GroupDef, 'SAP2000v1.cGroup');
s2k.GroupDef = GroupDef;
```

Figura Anexo1 7 Menú de archivo y de definiciones

Le siguen los menús de propiedades, cuyas definiciones son útiles cuando se requiera modificar la propiedad de cualquier elemento y posteriormente las definiciones de objetos, que engloban los diferentes tipos de objeto que aparecen en el programa como son cables, tendones, vigas, puntos, áreas etc.

```
% Properties menu
PropArea = NET.explicitCast(SapModel.PropArea, 'SAP2000v1.cPropArea');
PropFrame = NET.explicitCast(SapModel.PropFrame, 'SAP2000v1.cPropFrame');
PropSolid = NET.explicitCast(SapModel.PropSolid, 'SAP2000v1.cPropSolid');
PropMaterial = NET.explicitCast(SapModel.PropMaterial, 'SAP2000v1.cPropMaterial');
PropLink = NET.explicitCast(SapModel.PropLink, 'SAP2000v1.cPropLink');
PropCable = NET.explicitCast(SapModel.PropCable, 'SAP2000v1.cPropCable');
PropTendon = NET.explicitCast(SapModel.PropTendon, 'SAP2000v1.cPropTendon');
s2k.PropArea = PropArea;
s2k.PropFrame = PropFrame;
s2k.PropMaterial = PropMaterial;
s2k.PropLink = PropLink;
s2k.PropCable = PropCable;
s2k.PropTendon = PropTendon;
s2k.PropSolid = PropSolid;

% Objects menu
AreaObj = NET.explicitCast(SapModel.AreaObj, 'SAP2000v1.cAreaObj');
CableObj = NET.explicitCast(SapModel.CableObj, 'SAP2000v1.cCableObj');
FrameObj = NET.explicitCast(SapModel.FrameObj, 'SAP2000v1.cFrameObj');
LinkObj = NET.explicitCast(SapModel.LinkObj, 'SAP2000v1.cLinkObj');
PointObj = NET.explicitCast(SapModel.PointObj, 'SAP2000v1.cPointObj');
SolidObj = NET.explicitCast(SapModel.SolidObj, 'SAP2000v1.cSolidObj');
TendonObj = NET.explicitCast(SapModel.TendonObj, 'SAP2000v1.cTendonObj');
s2k.AreaObj = AreaObj;
s2k.CableObj = CableObj;
s2k.FrameObj = FrameObj;
s2k.LinkObj = LinkObj;
s2k.PointObj = PointObj;
s2k.SolidObj = SolidObj;
s2k.TendonObj = TendonObj;
```

Figura Anexo1 8 Menú de propiedades y de objetos

Después se han añadido las definiciones útiles para todos los casos de carga. Un menú muy utilizado ya que alguno de los casos de carga es necesario siempre en cualquier modelo para realizar el análisis. A este le sigue otro menú necesario siempre que es el de análisis que está compuesto únicamente de un concepto.

```
% Cases menu
LoadCases = NET.explicitCast(SapModel.LoadCases, 'SAP2000v1.cLoadCases');
Buckling = NET.explicitCast(LoadCases.Buckling, 'SAP2000v1.cCaseBuckling');
ModalEigen = NET.explicitCast(LoadCases.ModalEigen, 'SAP2000v1.cCaseModalEigen');
Moving = NET.explicitCast(LoadCases.Moving, 'SAP2000v1.cCaseMovingLoad');
StaticLinear = NET.explicitCast(LoadCases.StaticLinear, 'SAP2000v1.cCaseStaticLinear');
StaticNonlinear = NET.explicitCast(LoadCases.StaticNonlinear, 'SAP2000v1.cCaseStaticNonlinear');
SteadyState = NET.explicitCast(LoadCases.SteadyState, 'SAP2000v1.cCaseSteadyState');
s2k.LoadCases = LoadCases;
s2k.Buckling = Buckling;
s2k.ModalEigen = ModalEigen;
s2k.Moving = Moving;
s2k.StaticLinear = StaticLinear;
s2k.StaticNonlinear = StaticNonlinear;
s2k.SteadyState = SteadyState;

% Analyze menu
Analyze = NET.explicitCast(SapModel.Analyze,'SAP2000v1.cAnalyze');
s2k.Analyze = Analyze;
```

Figura Anexo1 9 Menú de casos y de análisis

Continua con los conceptos necesarios para los resultados entre los que aparecen dos conceptos utilizados para conservar los resultados los dependiendo del tipo de acción utilizada utilizarán un tipo de *array* u otro. Estos arrays pueden ser *double* o *string*.

```
% Resultados menu
AnalysisResults = NET.explicitCast(SapModel.Results,'SAP2000v1.cAnalysisResults');
AnalysisResultsSetup = NET.explicitCast(AnalysisResults.Setup,'SAP2000v1.cAnalysisResultsSetup');
SD1 = NET.createArray('System.Double', 1);
SS1 = NET.createArray('System.String', 1);

s2k.AnalysisResults = AnalysisResults;
s2k.AnalysisResultsSetup = AnalysisResultsSetup;
s2k.SD = SD1;
s2k.SS = SS1;
%handle.SL1 = SL1;
```

Figura Anexo1 10 Menú de resultados

Le siguen las definiciones de los tipos de objetos y elementos los cuales se definen de una forma distinta a todos los anteriores, igualando el término s2k.____ a un término SAP2000v1.eItemType._____ o SAP2000v1.eItemTypeElm._____ dependiendo el caso. Tras ello se iguala el término ok a 0 lo que supondrá que la función de inicialización ha funcionado correctamente. Finalmente, se pone un *end* que señala el final de la función.

```
% Tipos de objetos
s2k.Objects = SAP2000v1.eItemType.Objects;
s2k.Group = SAP2000v1.eItemType.Group;
s2k.SelectedObjects = SAP2000v1.eItemType.SelectedObjects;

% Elementos
s2k.ObjectElm = SAP2000v1.eItemTypeElm.ObjectElm;
s2k.Element = SAP2000v1.eItemTypeElm.Element;
s2k.GroupElm = SAP2000v1.eItemTypeElm.GroupElm;
s2k.SelectionElm = SAP2000v1.eItemTypeElm.SelectionElm;

ok = 0;
end
```

Figura Anexo1 11 Tipos de objetos y elementos

Script principal

Inicialización

Lo primero de todo es utilizar los comandos *clc* y *clear* para limpiar tanto el *workspace* como la ventana de comandos. Después se añade una variable *show* para indicar si se quiere que se abra la ventana de SAP2000®, en el caso de igualarse a *true*, o no, si se iguala a *false*. Tras ello, se incluye una variable con la ruta del fichero .dll de SAP2000®, que suele estar en la misma dirección solo que el nombre de la carpeta puede cambiar con cada versión.

```
clear;
clc;
% Un indicador de si quieres (true) o no (false) ver la ventana de SAP2000
show = true; %OJOOO
% Y la ruta al fichero .dll que tiene las funciones de la API. Suele estar
% en la ruta de instalación del programa y su nombre puede cambiar entre
% versiones.
APIPATH = 'C:\Program Files\Computers and Structures\SAP2000 22\SAP2000v1.dll';
```

Figura Anexo1 12 Inicialización del modelo

Posteriormente se ejecuta la función de inicialización con la ruta marcada y la intención de ver o no la ventana de SAP2000®. Por su parte, la función retorna un 0 si todo está bien y una variable con la instancia de SAP2000® y toda su funcionalidad. Además, se incluyen dos variables que pueden ser necesarias más adelante.

```
% Finalmente, se lanza la funcion initSAPv22.m
[ok, s2k] = initSAPv22(show, APIPATH);
% La función retorna dos argumentos:
% - ok: indica si ha ido todo bien (= 0) o no (> 0).
% - s2k: la variable con la instancia de SAP2000 y toda su funcionalidad.

% Además, renombramos un par de variables que nos van a venir bien más
% adelante:
SD = s2k.SD; % Double placeholder
SS = s2k.SS; % String placeholder
```

Figura Anexo1 13 Ejecución de la función de inicialización en el modelo

Abrir un modelo

Para abrir un modelo es necesario en primer lugar crear una variable con la ruta del modelo y usar el comando para abrir el modelo.

```
% Definir una variable con la ruta al modelo:
modelpath = './blanco_k_modif_v2.sdb';

s2k.File.OpenFile(modelpath);
% En este momento, todas las operaciones que se realicen a continuación se
% hacen sobre el modelo que acabamos de abrir. Como se aprecia, esa función
% también retorna un valor, "ok", que indica si ha ido todo bien (= 0) o no
% (> 0 ). En realidad, todas las funciones de la API de SAP2000 lo hacen.
```

Figura Anexo1 14 Definición de la ruta del modelo de SAP2000 y ejecución para abrirlo

Modificar rigidez

Cuando se trata de modificar una propiedad del modelo se realiza a través de una función *Set*. El tipo de funciones *Set* permiten asignar distintas propiedades o características al modelo dependiendo de la función utilizada.

Para este caso, que es la modificación de una rigidez, se realiza poniendo un muelle con una k distinta. La función necesaria para realizarlo es *SetSpring*. Acudiendo al archivo de ayuda esta función se encuentra en *CSi OAPI Functions* > *Object Model* > *Point Object* y obtenemos lo siguiente:

Syntax

SapObject.SapModel.PointObj.SetSpring

VB6 Procedure

Function SetSpring(ByVal Name As String, ByRef k() As Double, Optional ByVal ItemType As eItemType = object, Optional ByVal IsLocalCSys As Boolean = False, Optional ByVal Replace As Boolean = False) As Long

Figura Anexo1 15 Sintaxis de la función SetSpring

Aunque el lenguaje que viene por defecto en la ayuda es VB6, podemos ver el procedimiento con la información necesaria para modificar la rigidez:

- **Name**: el nombre del punto en el cual se quiere añadir/reemplazar el muelle. El punto ha de darse como tipo *String*, es decir, entre comillas simples como se ve más adelante en la Figura Anexo1 16.
- **k()**: la rigidez la cual se mete como *Double*. Las distintas posiciones del valor indican las diferentes componentes de la rigidez: [U1, U2, U3, R1, R2, R3], en las que U=[F/L] y R=[F·L/rad].
- **ItemType**: hay que introducir también el tipo de elemento. Cada elemento lleva asociado un número, Objecto = 0, Grupo =1 y Objetos Seleccionados = 2. Es opcional añadirlo, ya que si no se pone toma por defecto que es de tipo objeto.
- **IsLocalCSys**: se utiliza para indicar el sistema de referencia y se introduce con operaciones lógicas (booleanas). *True* para sistema local y *False* para sistema global. También es opcional. En caso de no ponerse toma por defecto que es *False*.
- **Replace**: en este caso también se utilizan las operaciones lógicas y sirve para especificar si el muelle que se va a añadir reemplaza a alguno que hubiera antes (*True*)

o se añade a mayores (*False*). Al igual que el anterior, toma que es *False* si no se especifica.

```
%Poner el modelo blanco con la rigidez modificada (envejecida)
point = '2';
k = [0, 0, k_bl_env, 0, 0, 0];
elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
isLocalSys = false; % false: global; true: local
replace = true; % true: sustituye; false: suma
s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);
```

Figura Anexo1 16 Modificación de la rigidez de los apoyos elásticos

Añadir masa

A la hora de añadir una masa también se realiza a través de una función *Set*. En este caso la función es *SetMass*:

Syntax

SapObject.SapModel.PointObj.SetMass

VB6 Procedure

Function SetMass(ByVal Name As String, ByRef m() As Double, Optional ByVal ItemType As eItemType = object, Optional ByVal IsLocalCSys As Boolean = True, Optional ByVal Replace As Boolean = False) As Long

Figura Anexo1 17 Sintaxis de la función SetMass en un punto

El procedimiento es muy similar al del caso anterior:

- **Name**: el nombre del punto en el cual se quiere añadir/reemplazar una masa. Es igual que en el ejemplo anterior, el punto ha de darse como tipo *String*.
- **m()**: la masa se introduce como tipo *Double*. Las distintas posiciones del valor indican las diferentes componentes de la masa: [U1, U2, U3, R1, R2, R3], en las que U=[M] y R=[M·L^2].
- **ItemType**: hay que introducir también el tipo de elemento. Cada elemento lleva asociado un número, Objecto = 0, Grupo =1 y Objetos Seleccionados = 2. De carácter opcional. Si no se dice lo contrario se toma como tipo objeto.
- **IsLocalCSys**: se utiliza para indicar el sistema de referencia y se introduce con operaciones lógicas (booleanas). *true* para sistema local y *false* para sistema global. Si no se especifica es *false*.
- **Replace**: en este caso también se utilizan las operaciones lógicas y sirve para especificar si el muelle que se va a añadir reemplaza a alguno que hubiera antes (*true*) o se añade a mayores (*false*). También es opcional así que cuando no se dice nada entiende que es *false*.

```
%Poner el modelo blanco con la masa modificada (envejecida)
  % Masa en el punto 14
    point = '14'; % Punto en que se aplica la carga
    mass = [ml4, ml4, ml4, 0, 0, 0]; % Vector de fuerzas/momentos
    elemType = s2k.Objects;
    localCSys = false;
    replace = true;
    s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);
```

Figura Anexo1 18 Aplicar una masa en un punto de la plataforma

Analizar un modelo

Para realizar el análisis de un modelo se usan las funciones que se encuentran en *CSi OAPI Functions* > *Analyze* en el archivo de ayuda. En primer lugar, se usa la función *SetRunCaseFlag*:

Syntax

SapObject.SapModel.Analyze.SetRunCaseFlag

VB6 Procedure

Function SetRunCaseFlag(ByVal Name As String, ByVal Run As Boolean, Optional ByVal All As Boolean = False) As Long

Figura Anexo1 19 Sintaxis de la función SetRunCaseFlag

Se compone de tres parámetros:

- **Name**: el nombre del caso de carga del cual se quiere especificar su bandera. El caso ha de darse como tipo *String*.
- **Run**: esto especifica si se quiere ejecutar o no el caso de carga. Se utiliza las operaciones lógicas *true* si se quiere ejecutar o *false* si no se desea ejecutar el caso de carga seleccionado.
- **All**: es un parámetro opcional que se especifica con las operaciones lógicas (booleanas) *true* o *false*. En caso de ser *true*, se establece la bandera (ejecutarse o no) especificada en Run para todos los casos de carga existentes.

En los modelos de este trabajo utilizamos esta función primero para poner todos los casos en *Do Not Run* y después activar solamente los que se desean ejecutar.

Tras ello, se utiliza la función *SetActiveDOF* que se utiliza para especificar los grados de libertad a la hora de analizar el modelo.

Syntax

SapObject.SapModel.Analyze.SetActiveDOF

VB6 Procedure

Function SetActiveDOF(ByRef DOF() As Boolean) As Long

Figura Anexo1 20 Sintaxis de la función SetActiveDOF

- **DOF()**: en este caso solo es necesario especificar los grados de libertad. Se hace mediante las operaciones lógicas (booleanas) *true* o *false* en un formato *long*. En caso de usar *true* se activan.

Por último, para ejecutar el análisis del modelo se utiliza la función *RunAnalysis* la cual no requiere de la especificación de ningún parámetro.

Syntax

SapObject.SapModel.Analyze.RunAnalysis

VB6 Procedure

Function RunAnalysis() As Long

Figura Anexo1 21 Sintaxis de la función RunAnalysis

Con las tres funciones ya se puede ejecutar el análisis de cualquier modelo correctamente como en el caso de la siguiente Figura:

```
% Calcular el modelo blanco
caseName = '';
toRun = false;
allCases = true;
s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not run"

%Activar casos Modal y SteadyState
s2k.Analyze.SetRunCaseFlag('DEAD', false);
s2k.Analyze.SetRunCaseFlag('MODAL', false);
s2k.Analyze.SetRunCaseFlag('F800', false);
s2k.Analyze.SetRunCaseFlag('F14', true);

% Set DOFs to analyze
s2k.Analyze.SetActiveDOF([true false true false true false]);

% Run analysis
s2k.Analyze.RunAnalysis();
```

Figura Anexo1 22 Ejecución del análisis del modelo habiendo activado los casos de carga necesarios

Análisis de los resultados

Para realizar el análisis de los resultados se utilizan las funciones que se encuentran en *CSi OAPI Functions* > *Analysis Results*. En ella se encuentran dos apartados, *Analysis Results Setup* y *Results*. Se empieza por el primero, donde se encuentran las funciones para configurar el análisis de los resultados.

La primera función tras haber ejecutado el análisis del modelo es *DeselectAllCasesAndCombosForOutput*. Esta función deselecciona todos los casos de carga y combinaciones de carga. No hay parámetros asociados a esta función:

Syntax

SapObject.SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput

VB6 Procedure

Function DeselectAllCasesAndCombosForOutput() As Long

Figura Anexo1 23 Sintaxis de la función DeselecAllCasesAndCombosForOutput

Tras ello se selecciona el caso de carga que se requiere mediante la función *SetCaseSelectedForOutput*:

Syntax

SapObject.SapModel.Results.Setup.SetCaseSelectedForOutput

VB6 Procedure

Function SetCaseSelectedForOutput(ByVal Name As String, Optional ByVal Selected As Boolean = True) As Long

Figura Anexo1 24 Sintaxis de la función de SetCasesSelectedForOutput

En esta función sí que hay parámetros necesarios que especificar:

- **Name**: el nombre del caso de carga. El caso ha de darse como tipo *String*.
- **Selected**: parámetro opcional de tipo booleano. Con él se especifica si el caso de carga seleccionado es el que se va a usar para el análisis del resultado (*true*) o no (*false*). Si no se especifica este parámetro se toma *true* por defecto.

Como lo que se estudia en este trabajo es el estado estacionario y el tipo de caso de carga que se utiliza es *Steady State*, la función a usar es *SetOptionSteadyState*. Existen funciones para el resto de casos, como son el análisis modal o el estático no linear entre otros, pero en este caso solo se va a usar el del análisis del estado estacionario.

Syntax

SapObject.SapModel.Results.Setup.SetOptionSteadyState

VB6 Procedure

Function SetOptionSteadyState(ByVal Value As Long, ByVal SteadyStateOption As Long) As Long

Figura Anexo1 25 Sintaxis de la función SetOptionSteadyState

Para esta función solo es necesario especificar dos parámetros de tipo *long*:

- **Value**: se asignará 1 en caso de ser la envolvente y 2 para frecuencias.
- **SteadyStateOption**: se asignará 1 si se quiere en fase y fuera de fase, 2 si se quiere la magnitud y 3 para ambas.

Finalmente, se obtienen los resultados. Las funciones para realizarlo se encuentran en la segunda carpeta llamada *Results*. En el caso de este TFM se trabaja fundamentalmente con las funciones de respuesta en frecuencia, por lo que se ha optado por obtener la aceleración de un punto, aunque también podría haberse hecho con la velocidad o con el desplazamiento. Para ello se hace uso de la función *JointAcc*.

Syntax

SapObject.SapModel.Results.JointAcc

VB6 Procedure

Function JointAcc(ByVal Name As String, ByVal ItemTypeElm As eItemTypeElm, ByRef NumberResults As Long, ByRef Obj() As String, ByRef Elm() As String, ByRef LoadCase() As String, ByRef StepType() As String, ByRef StepNum() As Double, ByRef U1() As Double, ByRef U2() As Double, ByRef U3() As Double, ByRef R1() As Double, ByRef R2() As Double, ByRef R3() As Double) As Long

Figura Anexo1 26 Sintaxis de la función JointAcc

En este caso son varios los parámetros a definir:

- **Name**: el nombre del punto. Dicho punto ha de darse como tipo *String*.
- **ItemTypeElm**: hay que introducir también el tipo de elemento. Cada elemento lleva asociado un número, Objecto = 0, Elemento =1, Grupo = 2 y Selección = 3.
- **NumberResults**: de tipo *long* y define el número total de resultados devueltos por el programa.
- **Obj**: se trata de un *array* que incluye el nombre del punto asociado a cada resultado. Es de tipo *String*.

- **Elm**: también se trata de un *array* que incluye el nombre del elemento asociado a cada resultado. También es de tipo *String*.
- **LoadCase**: otro *array* en el que los elementos introducidos son de tipo *string* y que en este caso incluye el caso de carga de cada resultado.
- **StepType**: incluye el tipo de paso, si existe, para cada resultado. Tipo *string*.
- **StepNum**: incluye el número de paso, si existe, para cada resultado. Tipo *double*.
- **U1, U2, U3**: *arrays* dimensionales en los que se incluyen las aceleraciones de traslación en cada dirección. U = [L/s^2]. Tipo *double*.
- **R1, R2, R3**: *arrays* dimensionales en los que se incluyen las aceleraciones de rotación en cada eje. R = [rad/s^2]. Tipo *double*.

Todos esos datos se extraen de la aplicación y se almacenan. Después, se recogen los datos que se quieren utilizar, que en este caso es la aceleración del punto en U3, por lo que se iguala a una variable y se interpretan dichos datos que contienen una parte real y una imaginaria.

```
% Resultados caso estático lineal
s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);

data = u3.double; % Recojo los datos de salida de SAP2000
re = data(1:901); % Los interpreto
im = data(902:end);
frf_b = re + 1i*im;

f = (0:900)*10/900;
bla = abs(frf_b);

figure
    semilogy(f,bla)
```

Figura Anexo1 27 Obtención de los resultados para un caso de carga estático lineal en un punto concreto

Desbloquear y cerrar modelo

Para desbloquear el modelo tanto al final antes de cerrarlo, como después de cada análisis se hace a través de la función *SetModelIsLocked*. Esta se encuentra en *CSi OAPI Functions* > *General Functions* > *SapModel* en el archivo de ayuda.

Syntax

SapObject.SapModel.SetModelIsLocked

VB6 Procedure

Function SetModelIsLocked(LockIt as Boolean) As Long

Figura Anexo1 28 Sintaxis de la función SetModelIsLocked

- **LockIt**: es el único parámetro necesario para definir la función. Se hace mediante las operaciones lógicas *true*, para bloquear el modelo, y *false*, para desbloquear el modelo.

Por último, la función de cerrar el programa es conocida como *ApplicationExit*:

Syntax

SapObject.ApplicationExit

VB6 Procedure

Function ApplicationExit(ByVal FileSave As Boolean) As Long

Figura Anexo1 29 Sintaxis de la función ApplicationExit

Esta función se encuentra en en *CSi OAPI Functions* > *General Functions* > *SapObject* y mediante ella se consigue cerrar el programa de SAP2000®. Solo necesita un parámetro para definirla:

- **FileSave**: este parámetro define si quiere guardarse el archivo o no. Esto se hace mediante las booleanas *true* y *false*.

```
% Asegurar que el modelo se queda desbloqueado
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%%%%%%%%
% CERRAR EL PROGRAMA %
%%%%%%%%%%%%%%%%%%%%%%

s2k.SapObject.ApplicationExit(false);
```

Figura Anexo1 30 Desbloqueo del modelo de SAP2000 y cierre del programa

ANEXO II: MODELOS MATLAB®

Todos los scripts de MATLAB® se han realizado utilizando la opción de *Live Script* (.mlx) ya que de esta forma resulta más sencillo visualizar los distintos resultados y también es más cómodo a la hora de ejecutar el código por secciones.

Modelo 1. Determinación de la rigidez. Caso rigidez común en ambos apoyos elásticos.

```
%%%%%%%%%%%%%%%%%
% INICIALIZACIÓN %
%%%%%%%%%%%%%%%%%
clear;
clc;
% Un indicador de si quieres (true) o no (false) ver la ventana de SAP2000
show = true;
% Y la ruta al fichero .dll que tiene las funciones de la API. Suele estar
% en la ruta de instalación del programa y su nombre puede cambiar entre
% versiones.
APIPATH = 'C:\Program Files\Computers and Structures\SAP2000 21\SAP2000v1.dll';

% Finalmente, se lanza la funcion initSAPv22.m
[ok, s2k] = initSAPv22(show, APIPATH);
% La función retorna dos argumentos:
% - ok: indica si ha ido todo bien (= 0) o no (> 0).
% - s2k: la variable con la instancia de SAP2000 y toda su funcionalidad.

% Además, renombramos un par de variables que nos van a venir bien más
% adelante:
SD = s2k.SD; % Double placeholder
SS = s2k.SS; % String placeholder

%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODELO BLANCO ENVEJECIDO %
%%%%%%%%%%%%%%%%%%%%%%%%%%

%FORMA 1 (Abrir un modelo ya modificado en SAP2000)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%
    % ABRIR UN MODELO %
    %%%%%%%%%%%%%%%%%%

    % Definir una variable con la ruta al modelo:
    %modelpath = './blanco_k_modif_v2.sdb';

    %s2k.File.OpenFile(modelpath);
    % En este momento, todas las operaciones que se realicen a continuación se
    % hacen sobre el modelo que acabamos de abrir. Como se aprecia, esa función
    % también retorna un valor, "ok", que indica si ha ido todo bien (= 0) o no
    % (> 0 ). En realidad, todas las funciones de la API de SAP2000 lo hacen.

    %k_blanco = 19500; %Rigidez inicial de la estructura en su puesta en servicio

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
FORMA 2 (Abrir un modelo en SAP2000 y modificar sus propiedades desde MATLAB)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%
    % ABRIR UN MODELO %
    %%%%%%%%%%%%%%%%%%

    % Definir una variable con la ruta al modelo:
    modelpath = './blanco_k_modif_v2.sdb';

    s2k.File.OpenFile(modelpath);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % MODIFICACIÓN DE PROPIEDADES %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    k_blanco = 19500; %Rigidez inicial de la estructura en su puesta en servicio

    k_bl_env = 15125.8; %rigidez blanco envejecido

    %Poner el modelo blanco con la rigidez modificada (envejecida)
    point = '2';
    k = [0, 0, k_bl_env, 0, 0, 0];
    elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
    isLocalSys = false; % false: global; true: local
    replace = true; % true: sustituye; false: suma
    s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

    point = '5';
    k = [0, 0, k_bl_env, 0, 0, 0];
    elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
    isLocalSys = false; % false: global; true: local
    replace = true; % true: sustituye; false: suma
    s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% CALCULAR %
%%%%%%%%%%%%

% Calcular el modelo blanco
caseName = '';
toRun = false;
allCases = true;
s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not run"

%Activar casos Modal y SteadyState
s2k.Analyze.SetRunCaseFlag('DEAD', false);
s2k.Analyze.SetRunCaseFlag('MODAL', false);
s2k.Analyze.SetRunCaseFlag('F800', false);
s2k.Analyze.SetRunCaseFlag('F14', true);

% Set DOFs to analyze
```

```
s2k.Analyze.SetActiveDOF([true false true false true false]);

% Run analysis
s2k.Analyze.RunAnalysis();

%%%%%%%%%%%%%%%%%%%%%%%%
% RECUPERAR RESULTADOS %
%%%%%%%%%%%%%%%%%%%%%%%%

%Obtener los resultados de FRF en el nodo 14 donde se aplica el shaker

% Resultados caso estático lineal
s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);

data = u3.double; % Recojo los datos de salida de SAP2000
re = data(1:901); % Los interpreto
im = data(902:end);
frf_b = re + 1i*im;

f = (0:900)*10/900;
bla = abs(frf_b);

figure
    xlabel('Frecuencia [Hz]')
    ylabel('Amplitud [(m/s2)/N]')
    semilogy(f,bla)

% Dejar el modelo desbloqueado antes de seguir
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%
% MODELO GRIS %
%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% COMPARAR %
%%%%%%%%%%%%

% Bucle que 1) Abra gris, 2) Modifique la k, 3) Calcule 4) Saque resultados
% 5) Compare 6*) vuelva a empezar o salga del bucle

%%%%%%%%%%%%%%%
% 1) Abrir gris

% Es buena práctica definir una variable con la ruta al modelo:
modelpath = './gris_k_modif_v2.sdb';
```

```
% Y luego usar el siguiente comando:
s2k.File.OpenFile(modelpath);

%%%%%%%%%%%%%%
% Realizar bucle

%bucle de 1000 en 1000

for m = k_blanco:-1000:0 %Indicar precisión aquí / comienza en la rigidez de blanco original

    k1 = m;

    %%%%%%%%%%%%%%
    % 2) Modificar k

    % Rigidez de un apoyo: cambiar el spring de un punto
    point = '2';
    k = [0, 0, k1, 0, 0, 0];
    elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
    isLocalSys = false; % false: global; true: local
    replace = true; % true: sustituye; false: suma
    s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

    point = '5';
    k = [0, 0, k1, 0, 0, 0];
    elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
    isLocalSys = false; % false: global; true: local
    replace = true; % true: sustituye; false: suma
    s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

    %%%%%%%%%%%%
    % 3) Calcular

    % Set cases to run
    caseName = '';
    toRun = false;
    allCases = true;
    s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not run"

    %Solo necesario activar casos modal y steadstate
    s2k.Analyze.SetRunCaseFlag('DEAD', false);
    s2k.Analyze.SetRunCaseFlag('MODAL', false);
    s2k.Analyze.SetRunCaseFlag('F800', false);
    s2k.Analyze.SetRunCaseFlag('F14', true);

    % Set DOFs to analyze
    s2k.Analyze.SetActiveDOF([true false true false true false]);

    % Run analysis
    s2k.Analyze.RunAnalysis();

    %%%%%%%%%%%%%%%
    % 4) Sacar resultados
    % Resultados del caso estático lineal
    s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
```

```
s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);

data = u3.double;
re = data(1:901);
im = data(902:end);
frf_g = re + 1i*im;

%%%%%%%%%%%%%%%%
% 5) Comparar resultados

Figure
    xlabel('Frecuencia [Hz]')
    ylabel('Amplitud [(m/s2)/N]')
    semilogy(f, abs(frf_g))
    hold on
    semilogy(f, abs(frf_b))
    hold off

    gri = abs(frf_g);

% Calcular error
error=0; %inicialmente

for j = 1:1:901
     error=error+abs(gri(j)-bla(j));
end

% Calucular k
format long %para ver los decimales de la k si los hay

if error == 0 %En caso de obtener el caso exacto
    k_v = m %valor exacto de la rigidez envejecida (puede ocurrir en el 1er caso)

    disp(['La rigidez del modelo ahora es = ' num2str(k_v)])
    break

elseif m == k_blanco && error ~= 0 %Si 1er caso no es error = 0
    error_min = error %creamos la variable error_fin de cara al sig elseif
    k_v2 = m

elseif abs(error) < abs(error_min) % si el nuevo error es menor que el minimo anterior
    error_min = error
    k_v2 = m

else

end
```

```
    s2k.SapModel.SetModelIsLocked(false); %desbloquear antes de volver a realizar el bucle

end

if error ~= 0 %no error_min xq si error=0 en el 1er caso -> error_fin no estaría inicializada
    disp(['La rigidez del modelo más próxima es = ' num2str(k_v2)])
    disp(['Con un error de valor = ' num2str(error_min)])
end

% Asegurar que el modelo se queda desbloqueado
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%%%%%%%%
% CERRAR EL PROGRAMA %
%%%%%%%%%%%%%%%%%%%%%%
s2k.SapObject.ApplicationExit(false);
```

Modelo 2. Determinación de la rigidez. Caso rigideces distintas en los apoyos elásticos.

```
%%%%%%%%%%%%%%%%%
% INICIALIZACIÓN %
%%%%%%%%%%%%%%%%%
clear;
clc;
% Un indicador de si quieres (true) o no (false) ver la ventana de SAP2000
show = true;
% Y la ruta al fichero .dll que tiene las funciones de la API. Suele estar
% en la ruta de instalación del programa y su nombre puede cambiar entre
% versiones.
APIPATH = 'C:\Program Files\Computers and Structures\SAP2000 22\SAP2000v1.dll';

% Finalmente, se lanza la funcion initSAPv22.m
[ok, s2k] = initSAPv22(show, APIPATH);
% La función retorna dos argumentos:
% - ok: indica si ha ido todo bien (= 0) o no (> 0).
% - s2k: la variable con la instancia de SAP2000 y toda su funcionalidad.

% Además, renombramos un par de variables que nos van a venir bien más
% adelante:
SD = s2k.SD; % Double placeholder
SS = s2k.SS; % String placeholder

%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODELO BLANCO ENVEJECIDO %
%%%%%%%%%%%%%%%%%%%%%%%%%%

%FORMA 1 (Abrir un modelo ya modificado en SAP2000)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%
    % ABRIR UN MODELO %
    %%%%%%%%%%%%%%%%%%

    % Definir una variable con la ruta al modelo:
    %modelpath = './blanco_k_modif_v2.sdb';

    %s2k.File.OpenFile(modelpath);
    % En este momento, todas las operaciones que se realicen a continuación se
    % hacen sobre el modelo que acabamos de abrir. Como se aprecia, esa función
    % también retorna un valor, "ok", que indica si ha ido todo bien (= 0) o no
    % (> 0 ). En realidad, todas las funciones de la API de SAP2000 lo hacen.

    %k_blanco = 19500; %Rigidez inicial de la estructura en su puesta en servicio
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%FORMA 2 (Abrir un modelo en SAP2000 y modificar sus propiedades desde MATLAB)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%
    % ABRIR UN MODELO %
    %%%%%%%%%%%%%%%%%%
```

```
    % Definir una variable con la ruta al modelo:
    modelpath = './blanco_k_modif_v2.sdb';

    s2k.File.OpenFile(modelpath);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % MODIFICACIÓN DE PROPIEDADES %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    k_blanco = 19500; %Rigidez inicial de la estructura en su puesta en servicio

    k_bl_env1 = 17321.6; %rigidez blanco envejecido
    k_bl_env2 = 19500;

    %Poner el modelo blanco con la rigidez modificada (envejecida)
    point = '2';
    k = [0, 0, k_bl_env1, 0, 0, 0];
    elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
    isLocalSys = false; % false: global; true: local
    replace = true; % true: sustituye; false: suma
    s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

    point = '5';
    k = [0, 0, k_bl_env2, 0, 0, 0];
    elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
    isLocalSys = false; % false: global; true: local
    replace = true; % true: sustituye; false: suma
    s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% CALCULAR %
%%%%%%%%%%%%

% Calcular el modelo blanco
caseName = '';
toRun = false;
allCases = true;
s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not run"

%Activar casos Modal y SteadyState
s2k.Analyze.SetRunCaseFlag('DEAD', false);
s2k.Analyze.SetRunCaseFlag('MODAL', false);
s2k.Analyze.SetRunCaseFlag('F800', false);
s2k.Analyze.SetRunCaseFlag('F14', true);

% Set DOFs to analyze
s2k.Analyze.SetActiveDOF([true false true false true false]);

% Run analysis
s2k.Analyze.RunAnalysis();

%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% RECUPERAR RESULTADOS %
%%%%%%%%%%%%%%%%%%%%%%%

%Obtener los resultados de FRF en el nodo 14 donde se aplica el shaker

% Resultados caso estático lineal
s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);

data = u3.double; % Recojo los datos de salida de SAP2000
re = data(1:901); % Los interpreto
im = data(902:end);
frf_b = re + 1i*im;

f = (0:900)*10/900;
bla = abs(frf_b);

figure
    xlabel('Frecuencia [Hz]')
    ylabel('Amplitud [(m/s2)/N]')
    semilogy(f,bla)

% Dejar el modelo desbloqueado antes de seguir
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%
% MODELO GRIS %
%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% COMPARAR %
%%%%%%%%%%%%

% Bucle que 1) Abra gris, 2) Modifique la k, 3) Calcule 4) Saque resultados
% 5) Compare 6*) vuelva a empezar o salga del bucle

%%%%%%%%%%%%%%%
% 1) Abrir gris

% Es buena práctica definir una variable con la ruta al modelo:
modelpath = './gris_k_modif_v2.sdb';
% Y luego usar el siguiente comando:
s2k.File.OpenFile(modelpath);

%%%%%%%%%%%%%%%
% Realizar bucle
```

```
%bucle de 1000 en 1000

for m = k_blanco:-1000:0 %Indicar precisión aquí / comienza en la rigidez de blanco original

    for n = k_blanco:-1000:0

        k1 = m;
        k2 = n;

        %%%%%%%%%%%%%%%
        % 2) Modificar k

        % Rigidez de un apoyo: cambiar el spring de un punto
        point = '2';
        k = [0, 0, k1, 0, 0, 0];
        elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
        isLocalSys = false; % false: global; true: local
        replace = true; % true: sustituye; false: suma
        s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

        point = '5';
        k = [0, 0, k2, 0, 0, 0];
        elemType = s2k.Objects; % A qué se refiere point? Objeto, grupo, seleccion...
        isLocalSys = false; % false: global; true: local
        replace = true; % true: sustituye; false: suma
        s2k.PointObj.SetSpring(point, k, elemType, isLocalSys, replace);

        %%%%%%%%%%%%%%
        % 3) Calcular

        % Set cases to run
        caseName = '';
        toRun = false;
        allCases = true;
        s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not
run"

        %Solo necesario activar casos modal y steadstate
        s2k.Analyze.SetRunCaseFlag('DEAD', false);
        s2k.Analyze.SetRunCaseFlag('MODAL', false);
        s2k.Analyze.SetRunCaseFlag('F800', false);
        s2k.Analyze.SetRunCaseFlag('F14', true);

        % Set DOFs to analyze
        s2k.Analyze.SetActiveDOF([true false true false true false]);

        % Run analysis
        s2k.Analyze.RunAnalysis();

        %%%%%%%%%%%%%%%
        % 4) Sacar resultados
        % Resultados del caso estático lineal
        s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
        s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);
```

```
s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);

data = u3.double;
re = data(1:901);
im = data(902:end);
frf_g = re + 1i*im;

%%%%%%%%%%%%%%%%
% 5) Comparar resultados

Figure
     xlabel('Frecuencia [Hz]')
     ylabel('Amplitud [(m/s2)/N]')
    semilogy(f, abs(frf_g))
    hold on
    semilogy(f, abs(frf_b))
    hold off

    gri = abs(frf_g);

% Calcular error
error=0; %inicialmente

for j = 1:1:901
     error=error+abs(gri(j)-bla(j));
end

% Calucular k
format long %para ver los decimales de la k si los hay

m %para mostrar todos los resultados
n
error

if m == k_blanco && n == k_blanco %primera iteracion se crea error min
    error_min = error %creamos la variable error_fin de cara al sig elseif
    k_1 = m
    k_2 = n

elseif abs(error) < abs(error_min) % si el nuevo error es menor que el minimo anterior
    error_min = error
    k_1 = m
    k_2 = n

elseif abs(error) == abs(error_min)

    disp(['Se ha alcanzado el mismo error mínimo obtenido con otra combinación'])
    error_minb = error
    k_1b= m
```

```
            k_2b= n

        else

        end
        s2k.SapModel.SetModelIsLocked(false); %desbloquear antes de volver a realizar el
bucle
    end
end

if error ~= 0 %no error_min xq si error=0 en el 1er caso -> error_fin no estaría inicializada
    disp(['La rigidez más próxima del apoyo 1 en p2 es = ' num2str(k_1)])
    disp(['La rigidez más próxima del apoyo 2 en p5 es = ' num2str(k_2)])
    disp(['Con un error de valor = ' num2str(error_min)])
end

% Asegurar que el modelo se queda desbloqueado
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%%%%%%%%
% CERRAR EL PROGRAMA %
%%%%%%%%%%%%%%%%%%%%%%
s2k.SapObject.ApplicationExit(false);
```

Modelo 3. Determinación de la masa en un punto conocido

```
%%%%%%%%%%%%%%%%%%
% INICIALIZACIÓN %
%%%%%%%%%%%%%%%%%%
clear;
clc;
% Un indicador de si quieres (true) o no (false) ver la ventana de SAP2000
show = true;
% Y la ruta al fichero .dll que tiene las funciones de la API. Suele estar
% en la ruta de instalación del programa y su nombre puede cambiar entre
% versiones.
APIPATH = 'C:\Program Files\Computers and Structures\SAP2000 22\SAP2000v1.dll';

% Finalmente, se lanza la funcion initSAPv22.m
[ok, s2k] = initSAPv22(show, APIPATH);
% La función retorna dos argumentos:
% - ok: indica si ha ido todo bien (= 0) o no (> 0).
% - s2k: la variable con la instancia de SAP2000 y toda su funcionalidad.

% Además, renombramos un par de variables que nos van a venir bien más
% adelante:
SD = s2k.SD; % Double placeholder
SS = s2k.SS; % String placeholder

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODELO BLANCO ENVEJECIDO %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%FORMA 1 (Abrir un modelo ya modificado en SAP2000)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%
    % ABRIR UN MODELO %
    %%%%%%%%%%%%%%%%%%

    % Definir una variable con la ruta al modelo:
    %modelpath = './blanco_m_modif.sdb';

    %s2k.File.OpenFile(modelpath);
    % En este momento, todas las operaciones que se realicen a continuación se
    % hacen sobre el modelo que acabamos de abrir. Como se aprecia, esa función
    % también retorna un valor, "ok", que indica si ha ido todo bien (= 0) o no
    % (> 0 ). En realidad, todas las funciones de la API de SAP2000 lo hacen.

    %k_blanco = 19500; %Rigidez inicial de la estructura en su puesta en servicio

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%FORMA 2 (Abrir un modelo en SAP2000 y modificar sus propiedades desde MATLAB)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%
% ABRIR UN MODELO %
%%%%%%%%%%%%%%%%%%
```

```
% Definir una variable con la ruta al modelo:
modelpath = './blanco_m_modif.sdb';

s2k.File.OpenFile(modelpath);
% En este momento, todas las operaciones que se realicen a continuación se
% hacen sobre el modelo que acabamos de abrir. Como se aprecia, esa función
% también retorna un valor, "ok", que indica si ha ido todo bien (= 0) o no
% (> 0 ). En realidad, todas las funciones de la API de SAP2000 lo hacen.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODIFICACIÓN DE PROPIEDADES %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   m_bl_env = 50; %masa punto 14

   m_shk = 40; % masa del shaker en el punto 14

   m14 = m_bl_env + m_shk;

  %Poner el modelo blanco con la masa modificada (envejecida)
  % Masa en el punto 14
    point = '14'; % Punto en que se aplica la carga
    mass = [m14, m14, m14, 0, 0, 0]; % Vector de fuerzas/momentos
    elemType = s2k.Objects;
    localCSys = false;
    replace = true;
    s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

%%%%%%%%%%%
% CALCULAR %
%%%%%%%%%%%

% Calcular el modelo blanco
caseName = '';
toRun = false;
allCases = true;
s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not run"

%Activar casos Modal y SteadyState
s2k.Analyze.SetRunCaseFlag('DEAD', false);
s2k.Analyze.SetRunCaseFlag('MODAL', false);
s2k.Analyze.SetRunCaseFlag('F800', false);
s2k.Analyze.SetRunCaseFlag('F14', true);

% Set DOFs to analyze
s2k.Analyze.SetActiveDOF([true false true false true false]);

% Run analysis
s2k.Analyze.RunAnalysis();

%%%%%%%%%%%%%%%%%%%%%%%%
% RECUPERAR RESULTADOS %
%%%%%%%%%%%%%%%%%%%%%%%%

%Obtener los resultados de FRF en el nodo 14 donde se aplica el shaker
```

```
% Resultados caso estático lineal
s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);


data = u3.double; % Recojo los datos de salida de SAP2000
re = data(1:901); % Los interpreto
im = data(902:end);
frf_b = re + 1i*im;

f = (0:900)*10/900;
bla = abs(frf_b);

figure
    xlabel('Frecuencia [Hz]')
    ylabel('Amplitud [(m/s2)/N]')
    semilogy(f,bla)

% Dejar el modelo desbloqueado antes de seguir
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%
% MODELO GRIS %
%%%%%%%%%%%%%%%

%%%%%%%%%%%%
% COMPARAR %
%%%%%%%%%%%%

% Bucle que 1) Abra gris, 2) Modifique la k, 3) Calcule 4) Saque resultados
% 5) Compare 6*) vuelva a empezar o salga del bucle

%%%%%%%%%%%%%%%
% 1) Abrir gris

% Es buena práctica definir una variable con la ruta al modelo:
modelpath = './gris_m_modif.sdb';
% Y luego usar el siguiente comando:
s2k.File.OpenFile(modelpath);

%%%%%%%%%%%%%%%
% Realizar bucle

%bucle de masa

for m = 0:10:50 %Indicar intervalo aquí / comienza en masa 0 kg (sin la del shk que se añade
despues)
```

```
m1 = m + m_shk; %teniendo en cuenta la masa del shaker

%%%%%%%%%%%%%%%
% 2) Modificar masa

% Masa en el punto 14
point = '14'; % Punto en que se aplica la carga
mass = [m1, m1, m1, 0, 0, 0]; % Vector de fuerzas/momentos
elemType = s2k.Objects;
localCSys = false;
replace = true;
s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

%%%%%%%%%%%%
% 3) Calcular

% Set cases to run
caseName = '';
toRun = false;
allCases = true;
s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not run"

%Solo necesario activar casos modal y steadstate
s2k.Analyze.SetRunCaseFlag('DEAD', false);
s2k.Analyze.SetRunCaseFlag('MODAL', false);
s2k.Analyze.SetRunCaseFlag('F800', false);
s2k.Analyze.SetRunCaseFlag('F14', true);

% Set DOFs to analyze
s2k.Analyze.SetActiveDOF([true false true false true false]);

% Run analysis
s2k.Analyze.RunAnalysis();

%%%%%%%%%%%%%%%
% 4) Sacar resultados
% Resultados del caso estático lineal
s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);

data = u3.double;
re = data(1:901);
im = data(902:end);
frf_g = re + 1i*im;

%%%%%%%%%%%%%%%
```

```
    % 5) Comparar resultados

    figure
         xlabel('Frecuencia [Hz]')
        ylabel('Amplitud [(m/s2)/N]')
        semilogy(f, abs(frf_g))
        hold on
        semilogy(f, abs(frf_b))
        hold off

        gri = abs(frf_g);

    % Calcular error
    error=0; %inicialmente

    for j = 1:1:901
         error=error+abs(gri(j)-bla(j));
    end

    % Calucular m
    format long %para ver los decimales de la k si los hay

    if error == 0 %En caso de obtener el caso exacto
        m_v2 = m1;
        error_min = error; %quitando el break

        %m_v = m %valor exacto de la masa (puede ocurrir en el 1er caso)
        %disp(['La masa en el punto 14 del modelo ahora es = ' num2str(m_v)])
        %break

    elseif m == 0 && error ~= 0 %Si 1er caso no es error = 0
        error_min = error %creamos la variable error_fin de cara al sig elseif
        m_v2 = m1

    elseif abs(error) < abs(error_min) % si el nuevo error es menor que el minimo anterior
        error_min = error
        m_v2 = m1

    else

    end
    s2k.SapModel.SetModelIsLocked(false); %desbloquear antes de volver a realizar el bucle

end


%if error ~= 0 %no error_min xq si error=0 en el 1er caso -> error_fin no estaría inicializada
    disp(['La masa añadida en el pto 14 del modelo más próxima es = ' num2str(m_v2-40)])
    disp(['La masa del pto 14 (añadiendo la del shaker) más próxima es = ' num2str(m_v2)])
    disp(['Con un error de valor = ' num2str(error_min)])
%end

% Asegurar que el modelo se queda desbloqueado
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%%%%%%%%
```

```
% CERRAR EL PROGRAMA %
%%%%%%%%%%%%%%%%%%%%%
s2k.SapObject.ApplicationExit(false);
```

Modelo 4. Determinación de la masa y localización

```
%%%%%%%%%%%%%%%%%
% INICIALIZACIÓN %
%%%%%%%%%%%%%%%%%
clear;
clc;
% Un indicador de si quieres (true) o no (false) ver la ventana de SAP2000
show = true;
% Y la ruta al fichero .dll que tiene las funciones de la API. Suele estar
% en la ruta de instalación del programa y su nombre puede cambiar entre
% versiones.
APIPATH = 'C:\Program Files\Computers and Structures\SAP2000 22\SAP2000v1.dll';

% Finalmente, se lanza la funcion initSAPv22.m
[ok, s2k] = initSAPv22(show, APIPATH);
% La función retorna dos argumentos:
% - ok: indica si ha ido todo bien (= 0) o no (> 0).
% - s2k: la variable con la instancia de SAP2000 y toda su funcionalidad.

% Además, renombramos un par de variables que nos van a venir bien más
% adelante:
SD = s2k.SD; % Double placeholder
SS = s2k.SS; % String placeholder

%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODELO BLANCO ENVEJECIDO %
%%%%%%%%%%%%%%%%%%%%%%%%%%

%FORMA 1 (Abrir un modelo ya modificado en SAP2000)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%
    % ABRIR UN MODELO %
    %%%%%%%%%%%%%%%%%%

    % Definir una variable con la ruta al modelo:
    %modelpath = './blanco_masa_pos_modif.sdb';

    %s2k.File.OpenFile(modelpath);
    % En este momento, todas las operaciones que se realicen a continuación se
    % hacen sobre el modelo que acabamos de abrir. Como se aprecia, esa función
    % también retorna un valor, "ok", que indica si ha ido todo bien (= 0) o no
    % (> 0 ). En realidad, todas las funciones de la API de SAP2000 lo hacen.

    %k_blanco = 19500; %Rigidez inicial de la estructura en su puesta en servicio

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%FORMA 2 (Abrir un modelo en SAP2000 y modificar sus propiedades desde MATLAB)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%
% ABRIR UN MODELO %
%%%%%%%%%%%%%%%%%%
```

```
% Definir una variable con la ruta al modelo:
modelpath = './blanco_masa_pos_modif.sdb';

s2k.File.OpenFile(modelpath);
% En este momento, todas las operaciones que se realicen a continuación se
% hacen sobre el modelo que acabamos de abrir. Como se aprecia, esa función
% también retorna un valor, "ok", que indica si ha ido todo bien (= 0) o no
% (> 0 ). En realidad, todas las funciones de la API de SAP2000 lo hacen.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODIFICACIÓN DE PROPIEDADES %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ELIMINAR MASAS PREVIAS
for p = 1:1:153
       p_str = num2str(p);
       point1 = p_str;
       itemType = s2k.Objects;
       s2k.PointObj.DeleteMass(point1, itemType);
   end

%NUEVA MASA Y POSICION PARA LUEGO ENCONTRARLA

%EVITAR LAS POSICIONES DE LOS APOYOS (1, 6, 7 y 9) PORQUE GENERAN ERROR
   m_bl_env = 18; %masa en blanco_modif
   punto_A = '25'; %posicion en la que está en blanco_modif

   m_shk = 40; %masa del shaker

  %Poner el modelo blanco con la masa modificada (envejecida)

  if strcmp(punto_A,'14') %en el caso de ser pto 14 hay que añadir masa shaker al total

    m_bl_env_14 = m_bl_env + m_shk;

    point = punto_A; % Punto en que se aplica la carga
    mass = [m_bl_env_14, m_bl_env_14, m_bl_env_14, 0, 0, 0];
    elemType = s2k.Objects;
    localCSys = false;
    replace = true;
    s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

  else %Si es en otro punto hay que aplicar la del shaker en 14 y la otra en el punto elegido
    point = punto_A; % Punto en que se aplica la carga
    mass = [m_bl_env, m_bl_env, m_bl_env, 0, 0, 0];
    elemType = s2k.Objects;
    localCSys = false;
    replace = true;
    s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

    point = '14'; % Punto en que se aplica el shaker
    mass = [m_shk, m_shk, m_shk, 0, 0, 0];
    elemType = s2k.Objects;
    localCSys = false;
    replace = true;
```

```
    s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

    end

%%%%%%%%%%%
% CALCULAR %
%%%%%%%%%%%

% Calcular el modelo blanco
caseName = '';
toRun = false;
allCases = true;
s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not run"

%Activar casos Modal y SteadyState
s2k.Analyze.SetRunCaseFlag('DEAD', false);
s2k.Analyze.SetRunCaseFlag('MODAL', false);
s2k.Analyze.SetRunCaseFlag('F800', false);
s2k.Analyze.SetRunCaseFlag('F14', true);

% Set DOFs to analyze
s2k.Analyze.SetActiveDOF([true false true false true false]);

% Run analysis
s2k.Analyze.RunAnalysis();

%%%%%%%%%%%%%%%%%%%%%%
% RECUPERAR RESULTADOS %
%%%%%%%%%%%%%%%%%%%%%%

%Obtener los resultados de FRF en el nodo 14 donde se aplica el shaker

% Resultados caso estático lineal
s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

nodo = '14';
elemType = s2k.ObjectElm; % Un punto del dibujo
[ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
    s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
    SD, SD, SD, SD, SD, SD, SD);

data = u3.double; % Recojo los datos de salida de SAP2000
re = data(1:901); % Los interpreto
im = data(902:end);
frf_b = re + 1i*im;

f = (0:900)*10/900;
bla = abs(frf_b);

figure
    xlabel('Frecuencia [Hz]')
    ylabel('Amplitud [(m/s2)/N]')
```

```
    semilogy(f,bla)

% Dejar el modelo desbloqueado antes de seguir
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%
% MODELO GRIS %
%%%%%%%%%%%%%%

%%%%%%%%%%
% COMPARAR %
%%%%%%%%%%

% Bucle que 1) Abra gris, 2) Modifique la k, 3) Calcule 4) Saque resultados
% 5) Compare 6*) vuelva a empezar o salga del bucle

%%%%%%%%%%%%%%
% 1) Abrir gris

% Definir una variable con la ruta al modelo:
modelpath = './gris_masa_pos_modif.sdb';
% Y luego usar el siguiente comando:
s2k.File.OpenFile(modelpath);

%%%%%%%%%%%%%%
% Realizar bucle

%bucle posición

%i = 1:1:153;
%i = setdiff(i, 28:81);

i = 0:1:27; %solo se consideran los 27 primeros puntos
for pos = i %solo hay puntos de 1 a 27 y de 82 a 153

    %Eliminar masas anteriores
    for p = 1:1:153
       p_str = num2str(p);
       point1 = p_str;
       itemType = s2k.Objects;
       s2k.PointObj.DeleteMass(point1, itemType);
   end

    %bucle masa
    for m = 0:10:20 %Indicar precisión aquí / comienza en masa 0 kg

        m1 = m;

        %%%%%%%%%%%%%%
        % 2) Modificar masa

         if pos == 14 %en el caso de ser pto 14 hay que añadir masa shaker al total

            m1 = m1 + m_shk;
```

```
            point = num2str(pos); % Punto en que se aplica la carga
            mass = [m1, m1, m1, 0, 0, 0];
            elemType = s2k.Objects;
            localCSys = false;
            replace = true;
            s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

          else %Si es en otro punto hay que aplicar la del shaker en 14 y la otra en el punto
elegido
            point = num2str(pos); % Punto en que se aplica la carga
            mass = [m1, m1, m1, 0, 0, 0];
            elemType = s2k.Objects;
            localCSys = false;
            replace = true;
            s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);

            point = '14'; % Punto en que se aplica el shaker
            mass = [m_shk, m_shk, m_shk, 0, 0, 0];
            elemType = s2k.Objects;
            localCSys = false;
            replace = true;
            s2k.PointObj.SetMass(point, mass, elemType, localCSys,replace);
        end

        %%%%%%%%%%%%%
        % 3) Calcular

        % Set cases to run
        caseName = '';
        toRun = false;
        allCases = true;
        s2k.Analyze.SetRunCaseFlag(caseName, toRun, allCases); % Esto pone todos en "do not
run"

        %Solo necesario activar casos modal y steadstate
        s2k.Analyze.SetRunCaseFlag('DEAD', false);
        s2k.Analyze.SetRunCaseFlag('MODAL', false);
        s2k.Analyze.SetRunCaseFlag('F800', false);
        s2k.Analyze.SetRunCaseFlag('F14', true);

        % Set DOFs to analyze
        s2k.Analyze.SetActiveDOF([true false true false true false]);

        % Run analysis
        s2k.Analyze.RunAnalysis();

        %%%%%%%%%%%%%%%%
        % 4) Sacar resultados
        % Resultados del caso estático lineal
        s2k.AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput();
        s2k.AnalysisResultsSetup.SetCaseSelectedForOutput('F14', true);

        s2k.AnalysisResultsSetup.SetOptionSteadyState(2, 1);

        nodo = '14';
```

```
        elemType = s2k.ObjectElm; % Un punto del dibujo
        [ok, N, Obj, Elm, LC, ST, SN, u1, u2, u3, r1, r2, r3] = ...
            s2k.AnalysisResults.JointAcc(nodo, elemType, 0, SS, SS, SS, SS, ...
            SD, SD, SD, SD, SD, SD, SD);

        data = u3.double;
        re = data(1:901);
        im = data(902:end);
        frf_g = re + 1i*im;

        %%%%%%%%%%%%%%%%
        % 5) Comparar resultados

        figure
            xlabel('Frecuencia [Hz]')
            ylabel('Amplitud [(m/s2)/N]')
            semilogy(f, abs(frf_g))
            hold on
            semilogy(f, abs(frf_b))
            hold off

            gri = abs(frf_g);

        % Calcular error
        error=0; %inicialmente

        for j = 1:1:901
             error=error+abs(gri(j)-bla(j)); %antes error=error+gri(j)-bla(j);
        end

        pos %puestos para ver el resultado de cada iteracion
        m
        error

        % Calucular k
        format long %para ver los decimales de la k si los hay

        %if error == 0 %En caso de obtener el caso exacto
        %    m_v2 = m;
        %    error_min = error; %quitando el break
        %    punto_min = pos;

        if pos == 1 && m == 0 %Primer caso inicializamos el error minimo
            error_min = error %creamos la variable error_fin de cara al sig elseif
            m_v2 = m
            punto_min = pos

        elseif abs(error) < abs(error_min) %&& pos ~= 1 && pos ~= 6 && pos ~= 7 && pos ~= 9
% si el nuevo error es menor que el minimo anterior a excepcion de en los apoyos
            error_min = error
            m_v2 = m
            punto_min = pos

        else
```

```
        end
        s2k.SapModel.SetModelIsLocked(false); %desbloquear antes de volver a realizar el
bucle

    end
end

if error ~= 0 %no error_min xq si error=0 en el 1er caso -> error_fin no estaría inicializada
    disp(['El punto con error mínimo es = ' num2str(punto_min)])
    disp(['La masa más próxima es = ' num2str(m_v2)])
    disp(['Con un error de valor = ' num2str(error_min)])

end


% Asegurar que el modelo se queda desbloqueado
s2k.SapModel.SetModelIsLocked(false);

%%%%%%%%%%%%%%%%%%%%%%%
% CERRAR EL PROGRAMA %
%%%%%%%%%%%%%%%%%%%%%%%
s2k.SapObject.ApplicationExit(false);
```

ANEXO III: RESULTADOS CASO MASA Y LOCALIZACIÓN CON VARIAS FRF

Resultados obtenidos para el caso de 18 kg en el punto 15 midiendo las FRFs en distintos puntos de la estructura.

Punto 25 m=18kg		Error para FRFs en distintos puntos para una masa de 18 kg en el punto 25										
		Lateral 1							Centro		Lateral 2	
Pto.	M (kg)	L/8 Error FRF10	2L/8 Error FRF12	3L/8 Error FRF14	4L/8 Error FRF2	5L/8 Error FRF19	6L/8 Error FRF21	7L/8 Error FRF23	3L/8 Error FRF15	5L/8 Error FRF20	3L/8 Error FRF18	5L/8 Error FRF25
1	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	20	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
2	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6977	1,0582	0,9284	0,5212	0,9625	1,0954	0,7248	0,9270	0,9625	0,9268	0,9626
	20	0,6445	0,9652	0,8178	0,4165	0,9061	1,0635	0,7108	0,8164	0,9062	0,8163	0,9068
3	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6974	1,0576	0,9277	0,5211	0,9618	1,0949	0,7245	0,9262	0,9618	0,9261	0,9619
	20	0,6449	0,9660	0,8188	0,4178	0,9072	1,0644	0,7112	0,8175	0,9073	0,8173	0,9079
4	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,9536	1,5218	1,5159	1,1395	1,5153	1,5124	0,9491	1,5152	1,5153	1,5150	1,5150
	20	0,9536	1,5217	1,5168	1,1395	1,5152	1,5128	0,949	1,5152	1,5152	1,5150	1,5150
5	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6977	1,0582	0,9284	0,5218	0,9625	1,0953	0,7248	0,9270	0,9625	0,9268	0,9626
	20	0,6445	0,9652	0,8178	0,4166	0,9061	1,0635	0,7108	0,8164	0,9062	0,8163	0,9068
6	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	20	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
7	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	20	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
8	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,9536	1,5218	1,5169	1,1395	1,5153	1,5124	0,9491	1,5152	1,5153	1,5150	1,5150
	20	0,9536	1,5217	1,5168	1,1395	1,5152	1,5124	0,949	1,5152	1,5153	1,5150	1,5150
9	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	20	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
10	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6954	1,1554	1,2151	1,0807	1,1724	1,0867	0,6594	1,2501	1,1724	1,2500	1,1722
	20	0,5467	1,0009	1,1743	1,0090	0,854	0,7653	0,4582	1,1931	0,8542	1,1729	0,8554
11	0	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6958	1,1555	1,2512	1,0807	1,1725	1,0869	0,6595	1,2502	1,1782	1,2500	1,7225
	20	0,5535	1,0008	1,1742	1,0090	0,8542	0,7651	0,4581	1,1729	0,8544	1,1728	0,8557

12	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,5159	0,9252	1,0606	0,8909	0,7437	0,6651	0,3977	1,0594	0,7438	1,0593	0,7446
	20	0,6407	0,9901	0,9418	0,5819	0,7774	0,8974	0,6006	0,9366	0,7774	0,9394	0,7775
13	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,5458	0,9278	1,0605	0,8909	0,7439	0,6652	0,3977	1,0593	0,7439	1,0592	0,7447
	20	0,6401	0,9925	0,9413	0,5819	0,7768	0,8965	0,5999	0,9392	0,7768	0,9390	0,7768
14	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,5554	0,8796	0,8657	0,6553	0,8011	0,8134	0,515	0,8655	0,8010	0,8655	0,8008
	20	0,3914	0,6013	0,5611	0,3334	0,266	0,3011	0,2006	0,5636	0,2661	0,5639	0,2665
15	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,5570	0,8819	0,8682	0,6553	0,8023	0,8151	0,5162	0,8669	0,8023	0,8670	0,8021
	20	0,3935	0,6041	0,5661	0,3321	0,2682	0,3055	0,2037	0,5634	0,2683	0,5648	0,2687
16	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6958	1,1554	1,2512	1,0807	1,1724	1,0868	0,6594	1,2010	1,1724	1,2500	1,1722
	20	0,5541	1,0010	1,1743	1,0095	0,854	0,7653	0,4582	1,1731	0,8542	1,1729	0,8554
17	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,5491	0,9282	1,0606	0,8909	0,7437	0,6651	0,3977	1,0594	0,7438	1,0593	0,7445
	20	0,6407	0,9937	0,9418	0,5819	0,7774	0,8974	0,6006	0,9397	0,7774	0,9394	0,7774
18	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,5571	0,8820	0,8684	0,6553	0,8024	0,8151	0,5162	0,8673	0,8024	0,8663	0,8021
	20	0,3940	0,6048	0,5670	0,3321	0,2682	0,3058	0,2040	0,5654	0,2683	0,5617	0,2687
19	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,4491	0,7228	0,7347	0,5683	0,7303	0,7154	0,4451	0,7342	0,7298	0,7342	0,7293
	20	0,1134	0,1832	0,1877	0,1465	**0,1849**	0,1805	0,1119	0,1876	0,1856	0,1875	0,1864
20	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,4492	0,7230	0,7351	0,5839	0,7299	0,7156	0,4452	0,7344	0,7301	0,7343	0,7295
	20	**0,1130**	**0,1826**	**0,1872**	**0,1463**	0,185	**0,1798**	**0,1114**	**0,1871**	**0,1848**	**0,1870**	0,1858
21	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,3799	0,6595	0,7816	0,7617	0,7405	0,5941	0,3328	0,7811	0,7405	0,7810	0,7405
	20	0,7309	1,0951	0,9309	0,4542	1,0066	1,1606	0,7704	0,9294	1,0066	0,9292	1,0067
22	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,3799	0,6595	0,7817	0,7617	0,7403	0,5940	0,3326	0,7811	0,7403	0,7811	0,7402
	20	0,7302	1,0941	0,9303	0,4542	1,0059	1,1596	0,7699	0,9288	1,0059	0,9286	1,0060
23	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6432	1,0715	1,1683	1,0139	1,1298	1,0336	0,6274	1,1675	1,1297	1,1674	1,1295
	20	0,4596	0,7927	0,9274	0,89	0,8624	0,6859	0,3788	0,9267	0,8625	0,9266	0,8626
24	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6433	1,0701	1,1684	1,0138	1,1299	1,0337	0,6269	1,1676	1,1298	1,1675	1,1296
	20	0,4596	0,7922	0,9274	0,8901	0,8619	0,6853	0,3787	0,9267	0,8620	0,9267	0,8622
25	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,4491	0,7228	0,7350	0,5683	0,7298	0,7154	0,4451	0,7342	0,7298	0,7342	0,7298
	20	0,1134	0,1832	0,1877	0,1465	0,1856	0,1805	0,1119	0,1876	0,1856	0,1875	**0,1856**
26	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,3799	0,6595	0,7816	0,7617	0,7409	0,5943	0,3328	0,7811	0,7405	0,7810	0,7405

	20	0,7309	1,0951	0,9309	0,4542	1,0066	1,1606	0,7704	0,9293	1,0066	0,9292	1,0067
27	**0**	0,9537	1,5218	1,5169	1,1395	1,5153	1,5125	0,9491	1,5153	1,5153	1,5151	1,5151
	10	0,6432	1,0705	1,1683	1,0139	1,1298	1,0336	0,6268	1,1675	1,1297	1,1674	1,2947
	20	0,4596	0,7923	0,9273	0,89	0,8624	0,6859	0,3792	0,9267	0,8625	0,9266	0,8626