Universidad Central “Marta Abreu” de Las Villas

Facultad de Construcciones

Departamento de Ingeniería Civil

TRABAJO DE DIPLOMA

Título:

Calibración de Modelos Estructurales. Empleo de la matriz Jacobiana.

Autor: Lenier Yervilla Herrera.

Tutor: Dr. Ernesto Chagoyén Méndez

Ing. Leonardo Rodríguez González

2016-2017

DEDICATORIA

Este trabajo está dedicado a mis padres, mi hermano y Claudia.

A mi abuela donde quieras que estés.

AGRADECIMIENTOS

Mis agradecimientos están dirigidos a mis padres Marta y Félix, que siempre han estado y estarán ahí para mí.

A mi hermano Heikel, que, aunque esté lejos su apoyo y ayuda me sirvieron para estar hoy aquí.

A Claudia por su paciencia, ayuda incondicional y por creer en mí. A su familia muchas gracias.

A toda mi familia por su apoyo y preocupación.

A todo aquel que estuvo involucrado de una forma u otra para que saliera con éxitos este trabajo.

A mis tutores el Dr. Ernesto Chagoyén Méndez y el Ing. Leonardo Rodríguez González.

Gracias a todo aquel que en todo este tiempo se preocupó y me apoyó en los momentos más difíciles, en especial Rossana, el Dany, Lázaro y Reinier.

RESUMEN

La presente investigación está encaminada a estimar parámetros estructurales (calibración o actualización) de modelos mecánicos de puentes confeccionados en SAP2000 para aprovechar las ventajas de la implementación de los métodos de optimización que posee MATLAB. Para ello se utilizan los medios que brinda la interface OAPI (*CSi Open Application Programming Interface*) entre SAP2000 y MATLAB con la novedad de la inclusión de la matriz Jacobiana en el proceso.

Mediante la interpretación y procesamiento de los resultados de ensayos modales realizados al puente objeto de estudio se identifican los parámetros del comportamiento dinámico como las frecuencias de oscilación natural y los modos principales de oscilaciones propias. Esto posibilita, empleando técnicas de optimización programadas en MATLAB (Algoritmos Genéticos y de Minimización no Lineal con Restricciones), identificar ciertos parámetros estructurales: el Módulo de Elasticidad (E) o de Rigidez al Cortante (G), el Coeficiente de Poisson (μ), la Resistencia Específica del Hormigón (f´c) en los elementos de hormigón armado, el Módulo de Young (Módulo de Elasticidad o de Deformación); además para elementos metálicos el límite de fluencia (Fy) y la resistencia última (Fu). Reconocidos dichos parámetros se obtiene un modelo calibrado en SAP2000 que facilita su diagnóstico y evaluación bajo diferentes estados de carga, buscando que la respuesta del modelo sea aproximadamente igual a la de la estructura real.

Palabras Clave: Model Updating, optimización, calibración de modelos, matriz Jacobiana.

ABSTRACT

The aim of this research is estimate structural parameters (calibration or updating) of mechanical bridge models Made in SAP2000, to take advantage of the Implementation of MATLAB optimization methods. This is done using the tools provided by the OAPI interface (CSi Open Application Programming Interface) between SAP2000 and MATLAB with the novelty of the inclusion of the Jacobian matrix in the process.

By means of the interpretation and processing of the results of modal tests carried out to the bridge object of study the parameters of the dynamic behavior like the frequencies of natural oscillation and the main modes of own oscillations are identified. Using optimization techniques programmed in MATLAB (nonlinear least squares and genetic algorithms) it is possible identified other structural parameters like: Modulus of Elasticity (E), rigidity to the cutting (G), the Poisson Coefficient (μ) and the Concrete Specific Resistance (f'c) on reinforced concrete elements, Modulus of Young (Modulus of Elasticity or Deformation). Also for metallic elements: yield strength (Fy) and ultimate strength (Fu). A model calibrated in SAP2000 is obtained, which facilitates their diagnosis and evaluation under different load states, aiming for the model's response to be approximately the same as the actual structure once these parameters have been identified.

ÍNDICE

INTRODUCCIÓN

La respuesta dinámica de un puente bajo cargas de viento, sismos y otras cargas vivas depende de sus propiedades estructurales: la masa, la rigidez y el amortiguamiento. Aunque estas propiedades pueden ser modeladas usando sofisticados modelos analíticos, el comportamiento real solo puede llegar a ser verificado haciendo ensayos de vibración y obteniendo resultados que facilitan la identificación de características dinámicas (frecuencias naturales, relaciones de amortiguamiento y modos de vibración) las cuales sirven como base para validación y/o actualización de los modelos analíticos al proveer las propiedades reales de la estructura y sus condiciones límite.

El análisis modal es un proceso mediante el cual se describe una estructura en términos de sus propiedades dinámicas o parámetros modales, estos son, la frecuencia, el amortiguamiento y los modos de vibración, para todos los modos en el rango de frecuencias de interés. Todas las estructuras poseen frecuencias naturales y modos de vibración que dependen básicamente de la masa y de la rigidez de la estructura. En el diseño es necesario identificar estas frecuencias y conocer cómo afecta a la respuesta de la estructura cuando una fuerza actúa sobre la misma. El análisis modal es una herramienta eficiente para describir, comprender y modelar el comportamiento de las estructuras (Rodríguez, 2005).

Se estudia la estructura cuando se encuentra sometida a una excitación conocida, con el objetivo de obtener un modelo matemático del comportamiento dinámico de la estructura. El procedimiento consiste en la adquisición de datos, su análisis, y luego la determinación de todos los parámetros modales. Los parámetros modales son importantes porque describen las propiedades dinámicas inherentes a una estructura (Rodríguez, 2005).

La estimación de parámetros estructurales (calibración o actualización) de un modelo de una estructura real es el proceso que consiste en ajustar los parámetros de un modelo mecánico de dicha estructura, minimizando la diferencia entre los valores de la respuesta real obtenidos de mediciones experimentales y la respuesta del modelo de la misma (también es conocido como *Model Updating*) (Teughels, 2003). Este procedimiento está basado en modelos más acertados en términos de evaluación de cargas, detección de desperfectos y daños estructurales, rehabilitación, etc., facilitando la toma de decisiones acerca del comportamiento de dicha estructura ante situaciones de carga o estado de la misma que no puedan ser experimentalmente ensayados.

Contar con el modelo numérico confiable de una estructura es importante para realizar un adecuado análisis o diseño estructural. Sin embargo, una estructura real está sujeta a muchas variables que influyen en su comportamiento. Las variables que pueden intervenir en la respuesta de una estructura pueden ser el tipo de cargas, el comportamiento del material, las condiciones de frontera y las conexiones entre los diferentes elementos que la conforman.

El rango de variación de estas variables influye de manera significativa en la respuesta de la estructura por lo que encontrar los valores óptimos representa la confiabilidad de la respuesta estructural. El proceso de optimización posibilita obtener los valores óptimos para estas variables a través del planteamiento de una ecuación de error que debe ser minimizada (Sovero and Martel, 2014).

En el caso de la calibración de modelos de puentes se han ejecutado manualmente durante largo tiempo métodos tradicionales de prueba-error; en los que en muchas ocasiones no se tiene en cuenta determinadas variables que influyen sobre el comportamiento estructural de los mismos. En la actualidad con el desarrollo de softwares computacionales (SAP2000, CSi Bridge, ABAQUS, ANSYS, MATLAB) se han realizado investigaciones que utilizan los métodos de estimación de parámetros en modelos estructurales basados en algoritmos cuyo propósito es minimizar la diferencia entre la respuesta modelada y la respuesta medida en la estructura real (Andersen, 2013, Galvis, 2002, Liu, 2004, Afonso, 2016).

Los algoritmos de calibración utilizados pueden ser modificados con el objetivo de aumentar la velocidad de convergencia de estos. En este caso, con la incorporación de expresiones analíticas que contienen a la matriz Jacobiana debe disminuir o aumentar la velocidad porque aumenta el número de iteraciones a realizar, resaltando que puede presentar superior velocidad a estudios anteriores donde se hacía un largo procedimiento para calibrar. Esto permite operar de una forma más eficiente las tolerancias de las variables del *toolbox* de optimización de MATLAB, así como readaptar los factores para mejorar el grado de calibración alcanzado, tratando de que los residuos tengan similar importancia.

Para esta investigación vale resaltar la incorporación de la matriz Jacobiana en el algoritmo de calibración, dicha matriz está formada por las derivadas parciales de primer orden de la función objetivo que se va a minimizar.

La matriz Jacobiana es la transpuesta del gradiente de una función F(x) y al determinante de la misma se le conoce como Jacobiano. Este da información sobre el comportamiento

de F(x) en el punto x que se está analizando. Después de cada iteración del procedimiento de optimización para calibrar el modelo, estos parámetros son modificados, unido a esto hay que calcular en cada paso la matriz de sensibilidad Jacobiana (J) incluida en el método de optimización.

La función objetivo asumida es el vector residual entre los parámetros de respuesta medidos o determinados de forma experimental y los mismos parámetros extraídos de la respuesta del modelo. Estos parámetros son función de las invariantes de la modelación, jugando un papel importante las invariantes de geometría, material y condiciones de apoyo si fueran extraídos del análisis modal operacional. Este vector se compone de los residuos por frecuencias naturales, los residuos por formas modales y los residuos por curvaturas modales. La sintaxis se programa al final de la función de optimización creada en MATLAB para interactuar con el modelo mecánico del puente en SAP2000 gracias al empleo de la OAPI (*CSi Open Application Programming Interface)*. Este procedimiento finaliza cuando se haya alcanzado un alto grado de correlación entre ambas respuestas, la medida y la modelada (Afonso, 2016).

Como antecedentes a esta investigación en Cuba se cuenta con un grupo de trabajos en los cuales se emplea el procedimiento de calibración implementado en MATLAB para la estimación del módulo de deformación en puentes de ferrocarril, basándose en los resultados de pruebas dinámicas y de carga estática. Esta rutina fue utilizada para la determinación de la respuesta estática o dinámica del puente por el StaBIL-2.0, *toolbox* para el análisis por MEF (Método de Elementos Finitos) de estructuras en (González, 2015). También se realizó una investigación reciente sobre la calibración de puentes donde se aplicaba la OAPI SAP2000-MATLAB arrojando resultados satisfactorios en la actualización de modelos con la estimación del módulo de deformación, sin embargo, la dificultad radica en que es un proceso muy largo (Afonso, 2016).

Se puede decir que las técnicas utilizadas para la optimización buscan un conjunto de valores de las variables de diseño, que haga mínima una función objetivo y desempeñe simultáneamente una serie de restricciones que dependen de las mismas variables (Bacuilima, 2011).

Problema Científico

En nuestro país los puentes están bajo cargas de viento, sismos (en zonas orientales) y otras cargas móviles y vivas, lo que trae consigo un proceso de deterioro de sus propiedades y aparición de fallos que afectan el fin por el cual fueron diseñados. Por estos motivos se lleva a cabo un grupo de investigaciones para erradicar estos males y actuar a favor de diseños más adecuados y un monitoreo continúo de los daños.

Cómo elaborar un procedimiento automatizado para calibrar modelos estructurales de puentes, basado en ensayos modales, que emplee la interface OAPI de SAP2000 y los recursos de optimización de que dispone actualmente MATLAB incluyendo la matriz Jacobiana para realizar un pronóstico certero de la respuesta de la estructura real.

Objeto de Estudio

Los métodos para la estimación de parámetros estructurales (calibración y validación) de modelos de estructuras de puentes mediante ensayos modales (*Model Updating*).

Campo de Acción

La instrumentación combinada con la modelación mecánica y análisis estructural de puentes.

Hipótesis

Un procedimiento automatizado para la calibración de estructuras de puentes implementado en MATLAB, basado en los Métodos de Optimización estructural que incluya la matriz Jacobiana, permite desarrollar un modelo calibrado en SAP2000 de la estructura del puente objeto de estudio con la misma respuesta que la obtenida en la estructura real, logrando así la estimación correcta de los parámetros estructurales correspondientes al mismo, para la evaluación y diagnóstico de puentes con una mayor precisión.

Objetivo General

Evaluar la influencia del empleo de la matriz Jacobiana en el procedimiento para la calibración de modelos estructurales a través de la minimización de las diferencias entre la respuesta estructural del modelo y la determinada experimentalmente en la estructura estudiada, auxiliándose de interface OAPI de SAP2000 con MATLAB.

Objetivos Específicos

1. Valorar el estado del conocimiento sobre estimación de parámetros estructurales basada en ensayos modales, empleando la OAPI de SAP2000 y MATLAB incluyendo la matriz Jacobiana para la minimización de los residuos entre la respuesta modelada y medida.
2. Desarrollar un procedimiento computarizado de estimación de parámetros de modelos estructurales (calibración) de puentes, minimizando los residuos entre la respuesta modelada y medida, utilizando la interface de MATLAB y la OAPI de SAP2000 aplicando la matriz Jacobiana durante el proceso.
3. Aplicar el procedimiento implementado al menos a un caso de estudio de un puente y evaluar la influencia del empleo o no de la matriz Jacobiana.

Preguntas de investigación

¿Cuál es el estado actual del conocimiento sobre métodos empleados para la estimación de parámetros estructurales basada en ensayos modales y la inclusión de la matriz Jacobiana, para el caso de estructuras en general y puentes en específico?

¿Cómo elaborar un procedimiento automatizado de estimación de parámetros de modelos estructurales (calibración) de puentes que incluya la matriz Jacobiana empleando la OAPI de SAP2000-MATLAB?

¿Cómo influye en el caso de estudio de un puente la aplicación del procedimiento implementado, utilizando la matriz Jacobiana?

Tareas científicas

1. Establecer el nivel de los conocimientos actuales sobre el tema del trabajo y sintetizar los aspectos esenciales de los mismos, incluyendo los principales comandos o funciones de los *toolbox* de optimización con los que cuenta el MATLAB para resolver problemas de optimización lineales y no-lineales, incluyendo la matriz Jacobiana en el proceso.
2. Redacción del Capítulo I: Estado del conocimiento sobre calibración de modelos estructurales.
3. Formulación completa del problema de estimación de parámetros estructurales en modelos de puentes que consiste en:

a) Elaboración del algoritmo de interface entre MATLAB y SAP2000 desarrollado en el entorno de programación MATLAB incluyendo la matriz Jacobiana para crear el modelo estructural del puente objeto de estudio y obtener los parámetros de la respuesta dinámica modelada.

b) Elección e implementación de la función objetivo conveniente para ser aplicada a este problema.

4. Redacción del Capítulo II: Procedimiento para la calibración de modelos estructurales utilizando la interface OAPI de SAP2000-MATLAB incluida la matriz Jacobiana.
5. Aplicación del procedimiento elaborado al menos a un caso de estudio, donde se obtenga el modelo calibrado.
6. Análisis de los resultados obtenidos.
7. Redacción del capítulo III: Influencia de la aplicación a un caso de estudio del procedimiento automatizado para la calibración estructural utilizando o no el Jacobiano.
8. Formular y redactar las Conclusiones y Recomendaciones.

Novedad científica

Hasta el momento los investigadores del país efectúan el proceso de calibración de los modelos de forma manual, convirtiéndose en un proceso lento y agotador, que muchas veces no garantiza la fiabilidad estructural requerida. Los resultados obtenidos no son los más ajustados al comportamiento estructural real de los puentes por lo que el empleo de la OAPI y la matriz de sensibilidad o Jacobiana para realizar la calibración de puentes en Cuba es novedoso. La implementación en forma automatizada de un procedimiento basado en técnicas de optimización, con el empleo de técnicas de programación en MATLAB que incluya la matriz Jacobiana en el proceso, garantiza la obtención de modelos calibrados en SAP2000.

Valor metodológico

La creación de un procedimiento automatizado para obtener modelos calibrados en SAP2000 mediante la OAPI y el lenguaje de programación en MATLAB que incluyan la matriz Jacobiana en el proceso empleando técnicas de optimización revisten enorme valor metodológico pues da apertura a investigaciones que permiten el mejoramiento y adquisición de nuevos conocimientos en esta rama de la ingeniería civil en el país ya que crea un nuevo instrumento para el análisis de datos. Significa un acercamiento a los procedimientos y tecnologías de última generación que están cobrando auge en muchos países actualmente.

Valor práctico

El algoritmo desarrollado que incluye la matriz Jacobiana posibilita que el procedimiento de estimación de parámetros estructurales en puentes sometido a ensayos modales sea:

- Un procedimiento automatizado, rápido en la evaluación y diagnóstico de los modelos, perfeccionando la toma de decisiones.
- La base para realizar la calibración de los modelos.
- Confiable, disminuyéndose así el grado de incertidumbre. (Permite una aproximación a la realidad).

Relevancia social

El presente trabajo es un facilitador para la evaluación y diagnóstico de las estructuras lo cual es imprescindible para valorar el comportamiento bajo la acción de diversas cargas. Este proceso se convierte inmediatamente en un camino obligado en aras del perfeccionamiento de los métodos utilizados hasta el momento por los expertos en Cuba.

Organización del informe

El presente informe se estructura de esta manera:

- Resumen
- Introducción
- Capítulo I: Estado del conocimiento sobre calibración de modelos estructurales.
- Capítulo II: Procedimiento para la calibración de modelos estructurales utilizando la interface OAPI de SAP2000-MATLAB incluida la matriz Jacobiana.
- Capítulo III: Influencia de la aplicación a un caso de estudio del procedimiento automatizado para la calibración estructural utilizando o no el Jacobiano.
- Conclusiones
- Recomendaciones
- Bibliografía

Capítulo I: Estado del conocimiento sobre calibración de modelos estructurales

En el siguiente capítulo se abordará un conjunto de temas que dan respuesta a algunas interrogantes que surgen en la investigación llevada a cabo sobre la estimación de parámetros estructurales en puentes. En este caso se hace mención al proceso de optimización de parámetros modales y obtención de parámetros estructurales del modelo. Se hace una descripción de los softwares y sus utilidades en el proceso de actualización de los modelos mecánicos de los puentes, incluyendo la matriz Jacobiana en el proceso.

1.1 Introducción

La estimación de parámetros estructurales (*Model Updating*) basada en ensayos modales permite realizar la calibración de modelos y la validación de las estructuras. Actualmente se desarrollan nuevos procedimientos que son más precisos a la hora de utilizar herramientas y recursos de tecnología avanzada que admite satisfacer las necesidades en cuanto a este proceso dentro de la Ingeniería Civil. Además, existe un grupo de procedimientos disponibles que son confiables, pero con la incorporación de los métodos computacionales se mejora ampliamente el desarrollo de dicho proceso y se obtiene calidad y rendimiento a la hora de calibrar modelos mecánicos de estructuras.

Las pruebas de vibración son una práctica bien conocida para la identificación de daños de estructuras en la ingeniería civil. Los parámetros modales reales de una estructura se pueden determinar a partir de los datos obtenidos mediante pruebas utilizando métodos de identificación de sistemas. Mediante la comparación de estos parámetros modales medidos con los parámetros modales de un modelo numérico de la misma estructura en condiciones no dañadas, es posible la detección, localización y cuantificación de daños (Reynders et al., 2007).

Para obtener los parámetros modales hay que tener en cuenta la optimización, que no es más que un proceso matemático que consiste en encontrar una solución a un problema planteado mediante la definición de una ecuación que se conoce como función objetivo.

Los métodos de optimización dependen de las características de esta función y se dividen en dos grupos: a) optimización de funciones lineales, y b) optimización de funciones no lineales. Asimismo, la función objetivo depende de ciertas variables identificadas previamente en un análisis de sensibilidad y cuyo valor se desconoce. Estas variables están sujetas a restricciones que limitan la zona de búsqueda de la solución que está

definida por un rango determinado de valores donde es posible encontrar el valor óptimo (Sovero and Martel, 2014).

El método que se empleará para llevar a cabo la actualización del modelo estructural en este caso de investigación será mediante mínimos cuadrados el cual es un procedimiento para optimizar funciones no lineales que consiste en encontrar el mínimo de una función basada en una suma de cuadrados, ajustándose los valores o las variables de diseño con la utilización de la matriz Jacobiana, siendo esta de gran ayuda pues permite lograr mejores resultados.

La calibración de modelos tiene como propósito obtener los parámetros modales reales del puente estudiado tales como: las frecuencias de oscilación natural, los modos principales de oscilaciones libres o forzadas, y el amortiguamiento. Estos son la base para determinar la respuesta real de la estructura en fuerzas o desplazamientos y posteriormente la identificación de los demás parámetros estructurales como: la rigidez (E.I), el módulo de elasticidad (E) o de rigidez al cortante (G), el coeficiente de Poisson (μ) y la resistencia específica del hormigón (f´c) en el caso de elementos estructurales de hormigón armado o el módulo de Young (módulo de elasticidad o de deformación), el límite de fluencia (Fy), la resistencia última (Fu) para el caso de elementos metálicos, o la rigidez de los apoyos (Afonso, 2016).

La realización del modelo mecánico del puente se hace a partir de estos parámetros estructurales obtenidos, empleando softwares como SAP2000, ABAQUS o ANSYS, que a su vez utilizan el MEF. Dicho modelo estará implementado desde MATLAB para realizar su calibración empleando las técnicas de optimización disponibles en su *toolbox* (Afonso, 2016).

Para el caso de la utilización de las nuevas herramientas computacionales se debe tener en cuenta la OAPI SAP2000, que permite ampliar significativamente el uso de métodos de optimización incorporados a MATLAB para realizar dicho proceso de calibración llevado más allá del marco de análisis estructural establecido actualmente por otros métodos que se utilizan (Sextos and Balafas, 2011).

Por lo tanto, la obtención del modelo calibrado facilita que la toma de decisiones acerca del comportamiento estructural ante situaciones de carga o estado de la estructura que no puedan ser experimentalmente ensayados, esté basada en modelos más acertados en términos de evaluación de cargas, detección de desperfectos y daños estructurales, rehabilitación de estructuras, etc. (Afonso, 2016).

1.2 Optimización en la calibración de un modelo estructural

El proceso de optimización de parámetros modales consiste en minimizar el error que existe entre los parámetros modales numéricos obtenidos a través de un modelo matemático, y los parámetros modales experimentales alcanzados mediante un ensayo experimental. La optimización resuelve el problema planteado a través de la función objetivo, logrando así encontrar el punto adecuado dentro del proceso (Sovero and Martel, 2014).

La optimización se basa en obtener un resultado que satisfaga en diferentes aspectos el proceso de calibración como el tiempo de máquina o la precisión del resultado después de realizado el mismo, es decir, consiste en mejorar el funcionamiento y la calibración a través de distintas metodologías (Sequera and Solano, 2013).

Se puede decir que las técnicas de optimización buscan un conjunto de valores de las variables de diseño, que haga mínima o máxima una función objetivo y cumpla a la vez una serie de restricciones que dependen de las mismas variables (Bacuilima, 2011).

1.2.1 Optimización Estructural. Clasificaciones

La optimización estructural se basa en «minimizar el costo o minimizar el peso», de tal manera que satisface al diseño y sus requerimientos. Hoy día se puede observar que todos los métodos modernos para el diseño óptimo son tan abiertamente generales que se realiza el análisis con una amplia variedad de hipótesis de «carga», o «situaciones» de colapso. Esto se ha realizado gracias a que la computación en esta rama de la optimización reduce el tiempo en todos sus procesos analíticos (Cujia Meza, 2010).

En el siguiente Diagrama 1.1 se coloca las distintas clasificaciones de optimización estructural que aparecen dependiendo del problema de diseño:

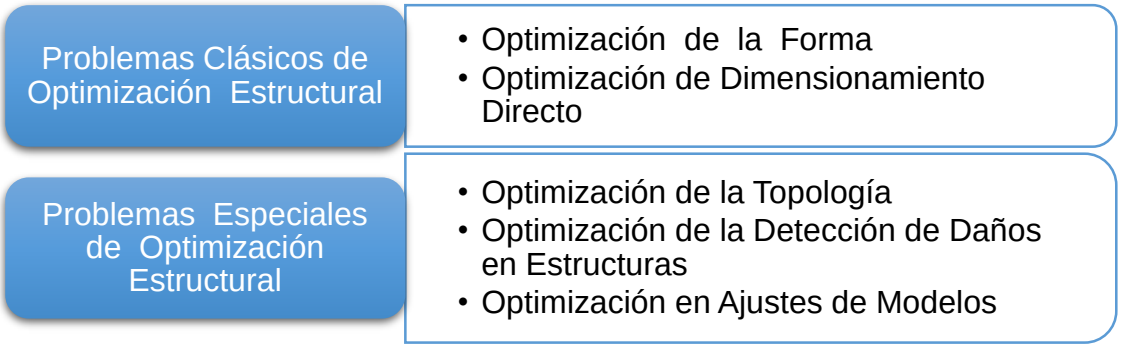


***Diagrama 1.1**: Tipos de Optimización Estructural según el tipo de problemas. Fuente: (Afonso, 2016).*

1.2.2 Conceptos básicos en un proceso de optimización

Los problemas de optimización generalmente están compuestos por variables, función objetivo y restricciones.

En el caso más simple, un problema radica en maximizar o minimizar una función real eligiendo sistemáticamente valores de entrada (tomados de un conjunto permitido) y computando el valor de la función.

La generalización de la teoría de la optimización y técnicas para otras formulaciones comprende un área grande de las matemáticas aplicadas. La optimización incluye el descubrimiento de los «mejores valores» de alguna función objetivo dado un dominio definido, incluyendo una variedad de diferentes tipos de funciones objetivo y tipos de dominios.

1.2.2.1 Variables de diseño

Las variables de diseño son variables independientes que son iteradas en orden para alcanzar el diseño óptimo, en ellas se especifican límites superiores e inferiores como «restricciones». Estos limitan el rango de variación de las Variables de Diseño (*DVs*, por sus siglas en inglés) (Rodríguez et al., 2007).

Algunos ejemplos de estas variables pueden ser el módulo de elasticidad, la densidad o el módulo de corte de un material. Hay que tener especial cuidado con los límites seleccionados debido a que estos son importantes en el momento de evaluar si un resultado es factible o no (Sequera and Solano, 2013).

1.2.2.2 Variables de estado

Variables de Estado (*SVs* por sus siglas en inglés) son cantidades que condicionan o restringen el diseño, se conocen también como «variables dependientes». Son típicamente variables de respuesta que son funciones de las variables de diseño. Una variable de estado puede tener un límite máximo y un límite mínimo, o puede tener un «único límite» (Rodríguez et al., 2007).

Algunos ejemplos de estas variables pueden ser las deflexiones en distintos puntos a lo largo de la geometría de un modelo o los modos de vibración y frecuencias naturales de una estructura (Sequera and Solano, 2013).

1.2.2.3 Función Objetivo

La función objetivo mide cuantitativamente el funcionamiento del sistema en un proceso de optimización, esto quiere decir que busca una maximización o minimización de esta. La función objetivo es una función vectorial de las variables de diseño y representa su propiedad más importante. Debido a la versatilidad de la función objetivo para adaptarse al problema propuesto, ella puede ser continua, discreta o mezclada en aquellos casos donde a partir de ciertos intervalos la función se define como discreta y en otros intervalos se define como continua (Cujia Meza, 2010).

Es una función de las *DVs*, entonces, si cambian los valores de las *DVs* cambia el valor de la función objetivo (Sequera and Solano, 2013). En este caso se debe trabajar tomando como función objetivo la ecuación (1) que incluye el vector residual $r(\theta)$ que se describe a continuación.

Para esta investigación la función objetivo queda definida de la siguiente forma (Reynders et al., 2007):

$$f(\theta) = \frac{1}{2}\|r(\theta)\|^2 \qquad (1)$$

$\theta =$ vector que contiene los parámetros a actualizar

$$r(\theta) = \begin{bmatrix} w_f r_f(\theta) \\ w_s r_s(\theta) \\ w_c r_c(\theta) \end{bmatrix} \qquad (1.1)$$

ω_f= factor de peso para residuos por frecuencias naturales

ω_s= factor de peso para residuos por formas modales

ω_c= factor de peso para residuos por curvatura modal

A su vez el vector residual consta de tres partes:

$$r_{fj}(\theta) = \frac{f_j(\theta)^2 - \widetilde{f_j^2}}{\widetilde{f_j^2}}$$ residuos por frecuencias naturales (1.1.1)

$$r_{sj}(\theta) = \emptyset_j(\theta) \cdot MSF_j - \widetilde{\emptyset_j}$$ residuos por formas modales (1.1.2)

$$r_{cj}(\theta) = k_j(\theta) \cdot MSF_j - \widetilde{k_j}$$ residuos por curvatura modal (1.1.3)

Los términos f_j y $\widetilde{f_j}$ denotan las frecuencias numéricas y experimentales del modo j, respectivamente.

Los términos $\emptyset_j$ y $\widetilde{\emptyset_j}$ denotan las formas modales numéricas y experimentales del modo j, respectivamente.

MSF_j es el factor de escala modal del modo j, definido en la ecuación 2.

$$MSF_j = \frac{\widetilde{\emptyset_j^T}\emptyset_j}{\emptyset_j^T\emptyset_j} \quad (2)$$

Los términos k_j y $\widetilde{k_j}$ denotan las curvaturas modales numéricas y experimentales del modo j, respectivamente. Este proceso es explicado también en (Afonso, 2016).

1.2.2.4 Restricciones

Las restricciones en el comportamiento del diseño estructural son generalmente las limitaciones en los esfuerzos o en los desplazamientos, también pueden tomar la forma de restricciones en frecuencias de vibración o probabilidad al colapso (Sequera and Solano, 2013). Poseen gran importancia dentro de la calibración de modelos pues permiten establecer un intervalo para el análisis del comportamiento de la estructura y determinar el daño. En este caso se tiene en cuenta limitaciones en la rigidez, módulo de elasticidad, período, etc., definiendo límites inferiores, superiores y en algunos casos pudieran emplearse restricciones de desigualdad.

1.2.3 Inclusión de la Matriz Jacobiana en la optimización del modelo estructural

1.2.3.1 Términos matemáticos

En términos matemáticos la matriz Jacobiana es una matriz formada por las derivadas parciales de primer orden de una función. Una de las aplicaciones más interesantes de esta matriz es la posibilidad de aproximar linealmente a la función en un punto. En este sentido, el Jacobiano representa la derivada de una función multivariable. Supongamos **F**: Rn → Rm es una función que va del espacio euclidiano n-dimensional a otro espacio euclidiano m-dimensional. Esta función está determinada por m funciones reales: y1(x1,..., xn),..., ym(x1,..., xn). Las derivadas parciales de estas (si existen) pueden ser organizadas en una matriz m-por-n, la matriz Jacobiana de ***F*** (Matemático, 2016).

Si *m* = *n*, entonces ***F*** es una función que va de un espacio n-dimensional a otro. En este caso la matriz Jacobiana es cuadrada y podemos calcular su determinante, conocido como el determinante Jacobiano o simplemente Jacobiano.

El determinante Jacobiano en un punto dado nos da información importante sobre el comportamiento de ***F*** cerca de ese punto. Para empezar, una función ***F*** es invertible cerca de **p** si el determinante Jacobiano en **p** es no nulo. Más aún, el valor absoluto del determinante en **p** nos da el factor con el cual ***F*** expande o contrae su volumen cerca de **p** (Matemático, 2016).

1.2.3.2 Matriz Jacobiana o Matriz de Sensibilidad en la programación

La matriz Jacobiana (J) o matriz de sensibilidad está formada por las derivadas parciales de primer orden de la función objetivo que se va a minimizar, siendo en este caso la ecuación (1) definida anteriormente.

Como se expresó previamente esta matriz contiene las derivadas de los elementos del vector residual *r* con respecto a los elementos del vector con los parámetros de rigidez θ. Esta matriz queda definida en la ecuación (3), que a su vez se aproxima por diferencias finitas (ecuación 3.1) (Reynders et al., 2007):

$$J(\theta) = \begin{bmatrix} \partial r1/\partial\theta 1 & \cdots & \partial r1/\partial\theta n \\ \vdots & \ddots & \vdots \\ \partial rm/\partial\theta 1 & \cdots & \partial rm/\partial\theta n \end{bmatrix} = [\partial r/\partial\theta 1 \quad \cdots \quad \partial r/\partial\theta n] \tag{3}$$

$$J(\theta) = [\Delta r/\Delta\theta 1 \quad \cdots \quad \Delta r/\Delta\theta n] \tag{3.1}$$

Si se utiliza una aproximación de diferencia creciente, la determinación de cada columna j en la ecuación (3.1) requiere de la determinación de los parámetros modales utilizados en r desde el modelo de elementos finitos para el cual el parámetro de rigidez j en cada columna es incrementado por $\Delta\theta_j$. Por lo tanto, se requiere de un número adicional de cálculos de elementos finitos igual al número de parámetros a actualizar en cada iteración. Sin embargo, estos cálculos pueden ser desarrollados por separado lo que posibilita un cálculo distribuido y una aplicación eficiente del método en los problemas a gran escala (Reynders et al., 2007).

Como se había explicado anteriormente, la función es invertible cerca de x (punto de análisis) si el Jacobiano en x es no nulo, además el valor absoluto del determinante en x da el factor con el cual la función se expande o contrae su volumen cerca de x. Con esta información para el caso del análisis estructural el método va buscando los valores donde el gradiente de la función cambia su signo lo cual indica que el punto óptimo está próximo al mismo (Bacuilima, 2011).

El problema mínimo no lineal de cuadrados es solucionado con una sensibilidad iterativa basada en el método de optimización utilizado. Por consiguiente, la matriz Jacobiana (o la matriz de sensibilidad) es calculada en cada iteración del proceso, por otra parte, el Jacobiano da información muy importante sobre el comportamiento de la función F(x) en el punto x que se está analizando para el caso de la estructura del puente.

Por cuestiones de comodidad para la programación en MATLAB se define θ=x donde θ es el vector que contiene los parámetros de rigidez (variables de diseño) a actualizar (calibrar), de los elementos correspondientes a las secciones en las que se divide el modelo del puente para su análisis. Es importante destacar la necesidad de que el número de parámetros a calibrar debe ser igual o menor que la cantidad de residuos evaluados en la función objetivo, para que el método *lsqnonlin* pueda ser utilizado (Afonso, 2016).

Después de cada iteración del procedimiento de optimización para calibrar el modelo estos parámetros son modificados, en conjunto hay que calcular en cada paso la matriz de sensibilidad Jacobiana J incluida en el método de optimización. La integración de este procedimiento de calibración explica según su forma de trabajo, que se basa en el correcto desempeño de la función creada en MATLAB (*lsqnonlin* es el que más se ajusta a esta investigación según (Reynders et al., 2007). Cuando la función objetivo es menor que la tolerancia adoptada, la función de optimización (*lsqnonlin*) termina su proceso y devuelve los valores buscados de los parámetros de rigidez (Afonso, 2016).

Los métodos de optimización basados en gradientes requieren trabajo computacional en el primer y segundo orden de sensibilidades de las predicciones del modelo con respecto a los parámetros modales, estas aparecen en el Jacobiano y la matriz Hessiana, respectivamente. Con este fin, dos avances básicos están disponibles, pero lo más popular es el uso del método de diferencias finitas que no es más que el método utilizado para calcular de manera aproximada las soluciones a las ecuaciones diferenciales usando ecuaciones diferenciales finitas para aproximar derivadas; sin embargo, este acercamiento consume mucho tiempo y es solo localmente exacto. Alternativamente, el uso de este procedimiento puede estar hecho de expresiones analíticas (Simoen et al., 2015).

1.2.4 Métodos de optimización para calibración de modelos estructurales

Todos los métodos de optimización involucran procesos matemáticos, por ende, según la clase a realizar se pueden clasificar en dos grandes grupos: Métodos Clásicos y Métodos Metaheurísticos (Cujia Meza, 2010).

Los métodos de optimización también pueden llamarse: Optimización Local (Clásico) y Optimización Global (Metaheurísticos). Los utilizados para la calibración de modelos de

elementos finitos para esta investigación son los de tipo local ya que los algoritmos que convergen más rápido resultan solo en una solución local.

Es importante tener en cuenta lo que es un mínimo local. Este no es más que un punto en el cual la función objetivo alcanza el valor más pequeño de los posibles, aunque los algoritmos de optimización local no siempre encuentran el mínimo absoluto (el mínimo global). El óptimo global es difícil de identificar porque el conocimiento de la función es usualmente solo local y el comportamiento de esta es desconocido en la región que no ha sido probada por el algoritmo (Teughels, 2003).

También los problemas de optimización pueden clasificarse en constreñidos y no constreñidos, dependiendo si se le imponen constricciones o no. En la optimización con constricciones las variables de diseño no pueden ser escogidas arbitrariamente, ellas más bien tienen que satisfacer algún requerimiento explícito como: limitaciones físicas en las variables formuladas por constricciones o relaciones entre las variables representadas por constricciones de igualdades o desigualdades lineales o no. Los problemas de optimización sin constricciones pueden emanar de algunas aplicaciones prácticas y jugar un rol central en la teoría de optimización, pues las optimizaciones con constricciones son, muchas veces, extensiones de algoritmos sin constricciones (Teughels, 2003).

1.2.4.1 Métodos Clásicos

Un método de optimización clásico es aquel que usa las derivadas (gradientes). Se puede decir que estos métodos buscan y garantizan un óptimo local, pero sin un tiempo determinado. Los métodos con derivadas se basan en tres algoritmos fundamentales: pendiente máxima (*steepest descent*), el método de Newton (*Newton's Method*) y el método *Levenberg-Marquardt* (Teughels, 2003).

Estos métodos están clasificados en tres categorías según otros autores. Se distingue los métodos directos de búsqueda (*Direct Search*), los de primer orden (*First Order*) y los de segundo orden (*Second Order*).

Los métodos directos de búsqueda usan sólo valores de la función objetivo. Los métodos de primer orden «también llamados los métodos de gradiente» requieren de los valores de las derivadas de primer orden de la función, son generalmente más eficientes cuando la función objetivo es continua en su primera derivada. El método de segundo orden, como el método de Newton, se halla sólo realmente en disposición cuando la información de segundo orden está fácilmente disponible y puede calcularse simplemente desde que el uso de la diferenciación numérica es computacionalmente valiosa (Teughels, 2003). La siguiente Tabla 1.1 hace un resumen de las categorías antes mencionadas.

	Line Search	***Trust Region***
Direct Search	*Coordinate Descent*	
	Hooke and Jeeves	
	Powell´s Method	
	Rosenbrock´s Method	
First-Order	*Quasi-Newton*	*Quasi-Newton*
	Steepest Descent	*Steepest Descent*
	Nonlinear Conjugate Gradient	
Second-Order	*Modified Newton Method*	*TR Newton Method*

Tabla 1.1: *Clasificación de los métodos. Fuente: (Teughels, 2003).*

El método básico es el método Newton Puro que tiene una tasa cuadrática de convergencia. Otros métodos son lo Quasi-Newton, conjugar gradientes, descenso de pendientes y los métodos de búsqueda directos libre de gradientes (Teughels, 2003). Estos nombres utilizados dependen principalmente del idioma que se emplee y las traducciones de los autores.

El método Newton Puro no tiene su convergencia garantizada, pero existen dos estrategias para implementarla, la búsqueda de la línea y la región de confianza. En lugar de calcular la matriz Hessiana exactamente, los métodos de Quasi-Newton lo aproximan usando sólo las primeras derivadas. Los métodos de gradiente no factorizan matrices eficazmente para solucionar problemas de gran escala. Para las funciones discontinuas, los métodos directos libre de gradientes de búsqueda pueden ser aplicados, pero son bastante lentos (Teughels, 2003).

Sin embargo, no son los únicos métodos y algoritmos, pero todos los demás resultan de mejoras y combinaciones de estos (Cujia Meza, 2010).

1.2.4.2 Métodos Metaheurísticos

Una metaheurística es un método heurístico para resolver un tipo de problema computacional general usando los parámetros dados por el usuario sobre unos procedimientos genéricos y abstractos de una manera que se espera sea eficiente (Blum and Roli, 2003). Un método heurístico es aquel que no utiliza metodología común y rigurosa para obtener un resultado.

De manera general estos métodos dan un buen resultado en tiempo aceptable, además de ser aplicables cuando se presenta una función objetivo con múltiples puntos óptimos o cuando estas funciones están compuestas por intervalos continuos y discretos. Se

fundamentan en el uso de conceptos intuitivos basados en sistemas naturales, como por ejemplo la evolución. El solo hecho de no usar derivadas que en ciertos casos pueden ser tediosas o aun casi imposibles de obtener, les da a estos métodos una flexibilidad, y una gran extensión de aplicabilidad cuando las condiciones del modelo son complejas.

La manera de encontrar la solución de dichos métodos es a través de la iteración. Algunos ejemplos de estos métodos son: Tomado de (Cujia Meza, 2010).

- Búsqueda Local
- Algoritmos Genéticos.
- Endurecimiento por calor Simulado (*Simulated Annealing SA*).
- Búsqueda Aleatoria (*Random Search*).
- Método Nelder–Mead o Método Ameba (*Downhill Simplex Search*).
- Optimización por Enjambres de Partículas (*Particle Swarm Optimization "PSO*").
- Ascenso de montañas (*Stochastic Hill-climbing*).
- Algoritmos Meméticos.

El objetivo de la optimización combinatoria es encontrar un objeto matemático finito que maximice o minimice, dependiendo del problema y la función especificada por el usuario (Blum and Roli, 2003).

Los métodos globales de búsqueda, como los algoritmos genéticos (*GA*) y simulado endureciendo por calor (*SA Simulated Annealing*), son en general más robustos, o sea la elección de la posición de puesta en marcha influye poco en el resultado final, y es más probable para detectar el mínimo global. *GA* actúa sobre una población de proposiciones de búsqueda y genera a las nuevas poblaciones. *SA* trata de evitar el mínimo local pero probabilísticamente tomando el valor menos óptimo en el espacio de búsqueda. El inconveniente principal de tales algoritmos es que requieren un número grande de evaluaciones de la función ya que se basan en la búsqueda probabilística sin el uso de cualquier información del gradiente. *GA* y *SA* son frecuentemente usados en la optimización estructural, donde el objetivo es lograr la optimización en el diseño de una estructura, minimizar el peso de la estructura o el costo de materiales (Teughels, 2003).

Recientemente, el método de minimizadores locales acoplados (*CLM Coupled Local Minimizers)* ha sido utilizado para los problemas globales de optimización. Debido a la información, el cambio y el uso simultáneo de proposiciones de búsqueda de múltiplo, el método puede encontrar el mínimo global. Además, una convergencia relativamente rápida es mantenida desde que el método se basa en las derivadas de la función en las proposiciones de búsqueda (Teughels, 2003).

1.2.5 Problema de mínimos cuadrados

Los métodos para solucionar tales problemas son iterativos y cada paso de iteración usualmente requiere la solución de un problema mínimo lineal de cuadrados. El problema mínimo no lineal de cuadrados es un caso especial del problema general de optimización (Teughels, 2003). El método básico de mínimos cuadrados es el Gauss-Newton, derivativo del método Newton para los problemas generales explicado en el epígrafe anterior.

En este caso se hace énfasis en el método Gauss-Newton, pero para problemas moderados o de grandes residuales o no lineales, la velocidad de convergencia del método del *Gauss-Newton* puede ser muy inferior al método de Newton (Teughels, 2003). Estos métodos son clasificados como mínimos cuadrados lineales y no lineales los cuales quedan resumidos en la siguiente Tabla 1.2:

Linear Least Squares	
	Cholesky factorization
	QR factorization
	Singular value decomposition
Nonlinear Least Squares	
First-Order	*Gauss-Newton*
	Levenberg-Marquardt
	Hybrid Quasi-Newton
	(e.g Hybrid Quasi- and Gauss-Newton)
Hybrid First- and Second-Order	*Hybrid Newton*
	(e.g Hybrid Newton- and Gauss-Newton)
Second-Order	*Newton*

Tabla 1.2: *Clasificación de los métodos. Fuente: (Teughels, 2003).*

En los problemas mínimos no lineales de cuadrados la función objetivo consta de una suma de cuadrados de funciones no lineales. El *Gauss-Newton* y los métodos de *Levenberg-Marquardt* aproximan la Hessiana usando solo la matriz Jacobiana. Este acercamiento es justificado para problemas de pequeños residuales o no lineales, en los otros casos, Newton híbrido o métodos de *Quasi-Newton* deberían ser aplicados (Teughels, 2003). Por esto en nuestra investigación se utilizan estos métodos, lo que puede ser mirado como un caso especial de un problema de optimización y donde son aplicados los métodos iterativos para solucionarlo.

1.2.6 Criterio de Correlación Modal (MAC)

Modal Assurance Criterion (MAC) criterio modal de seguridad o criterio de correlación modal para su traducción al español. Es un criterio que establece una relación entre dos vectores: el vector de desplazamientos modales obtenidos en los ensayos experimentales y el vector de desplazamientos modales obtenidos del modelo numérico que establece una relación entre dos vectores modales, extendiéndose desde 0-1. Mientras más cercano el valor de la MAC es para 1, más coherente son los resultados, lo que implica una correlación perfecta entre las formas modales del modelo numérico y las formas modales de la estructura real (Teughels, 2003, Sovero and Martel, 2014).

Este método proporciona una medida de la consistencia (grado de linealidad) entre estimaciones de un vector modal f, mediante la determinación de la desviación mínima cuadrada de una estimación sobre otra (Ancona et al., 2011).

MAC queda definida (Teughels, 2003):

$$MAC\ (\emptyset i, \emptyset j) = \frac{\left|\emptyset i^T \emptyset j\right|^2}{(\emptyset i^T \emptyset i)(\emptyset j^T \emptyset j)} \qquad (4)$$

Siendo $ø_i$ y $ø_j$ un modo numérico y modo experimental respectivamente, lo que permite formar el vector.

Para un modo experimental, el modo analítico correspondiente está definido como el modo analítico que enseña el valor más alto de la MAC en combinación con ese modo experimental. Ninguna expansión de las formas experimentales de modo es necesaria. Aunque es altamente efectiva para muchas estructuras, la MAC es poco confiable en la correlación de modos locales que se describe en (Teughels, 2003) por sólo muy pocos *DOFs*. También en caso de un número muy limitado de *DOFs* experimentales, un modo analítico puede tener valores comparables de la MAC para un número de modos experimentales. Sin embargo, de ser simple y algo intuitivo, el método puede ser muy ineficiente en la práctica, en el estado en que se encuentra para una función cuadrática en dos variables (Teughels, 2003).

Este criterio es un señalizador estadístico, justamente como la coherencia ordinaria, puede ser muy poderoso cuando es usado correctamente pero no es seguro cuando se usa incorrectamente.

La función del MAC es proveer una medida de consistencia (el grado de linealidad) entre las estimaciones de un vector modal. Esto provee un factor adicional de confianza en la

evaluación de un vector modal de posiciones diferentes de excitación (la referencia) o los algoritmos modales diferentes de estimación de parámetros (Allemang, 2003).

Se utiliza como una medida para la correspondencia entre una forma calculada de modo y una forma medida de modo. Como los valores de los parámetros varían, las formas de modo podrían traer diferencias, resultando en saltos bruscos en la función objetivo que puede impedir la eficiencia significativa de algoritmos basados en gradientes de optimización. Este criterio puede ayudar en una cantidad relativamente limitada de datos modales y un modelo simple de predicción, identificando, localizando, y cuantificando el daño estructural (Simoen et al., 2015).

Los usos del MAC quedan resumidos por (Allemang, 2003):

- La validación de modelos modales experimentales
- Correlación con modelos modales analíticos (*mode pairing*)
- Correlación con operación de vectores de respuesta
- Mapeo de matrices por medio analítico y experimental de modelos modales
- Análisis de error de vectores modales
- Promediado de vectores modales
- Terminación vectorial modal experimental y/o la expansión
- Algoritmos de actualización de modelos
- Consistencia/Estabilidad vectorial modal en los algoritmos modales de estimación de parámetros
- Detección de efecto/daño estructural
- Las evaluaciones de control de calidad
- La colocación óptima del sensor

En la actualidad este criterio ha sido modificado y se han creado otros nuevos: el criterio modal coordinado (COMAC) de seguridad, el criterio de seguridad de respuesta de frecuencia (FRAC), chequeo coordinado (CORTHOG) de ortogonalidad, criterio de frecuencia modal modificado a escala (FMAC) de seguridad, el criterio modal parcial (PMAC) de seguridad, criterio modal a escala (SMAC) de seguridad, y el criterio modal de seguridad usando vectores modales recíprocos (MACRV). El desarrollo de criterios de seguridad ha sido iniciado por defectos, reales o percibidos, del criterio modal original de seguridad MAC (Allemang, 2003).

Este criterio demuestra que un concepto estadístico simple puede convertirse en una herramienta sumamente útil en el campo de análisis modal experimental y dinámica estructural.

El uso del criterio modal de seguridad y el desarrollo y el uso de un número significante de criterios, han sido notables y son más probables debido a la simplicidad global del concepto (Allemang, 2003).

1.3 Obtención de parámetros modales. Generalidades

Cuando la respuesta dinámica de un modelo numérico es similar al de la estructura real se puede decir que dicho modelo numérico es confiable. La respuesta dinámica de una estructura está definida por tres parámetros modales: frecuencias de vibración, amortiguamiento y formas modales de vibración (Afonso, 2016).

El proceso de obtención de parámetros modales consiste en minimizar el error que existe entre los parámetros modales numéricos, obtenidos a través de un modelo matemático, y los parámetros modales experimentales de un ensayo experimental (Sovero and Martel, 2014).

Estos parámetros modales dependen de los valores de ciertas variables desconocidas que deben haber sido identificadas previamente en el análisis de sensibilidad. Por lo tanto, el proceso de optimización de parámetros modales tiene como objetivo encontrar los valores óptimos para estas variables desconocidas de tal manera que el error entre estos parámetros modales, analíticos y experimentales, se minimice. Para llevar a cabo este proceso se utiliza la herramienta de mínimos cuadrados no lineales, debido a las características de no linealidad de la función objetivo (Sovero and Martel, 2014).

Para optimizar funciones no lineales en base a mínimos cuadrados existen varios algoritmos y su elección depende de las características del problema a resolver. Cada algoritmo tiene su propio método de solución.

Por ejemplo, el algoritmo de *Gauss-Newton* es un sistema iterativo de cuadrados lineales que se aproximan al problema no lineal y tiene la ventaja de que para ser resuelto no es necesario hallar la segunda derivada (Hessiana) de la función (Sovero and Martel, 2014).

El proceso de optimización de parámetros modales se realiza a través de una ecuación llamada función objetivo o función error que relaciona las frecuencias y formas modales de vibración. Esta relación se efectúa planteando ecuaciones que dependan de las variables desconocidas identificadas en el análisis de sensibilidad (Sovero and Martel, 2014).

1.3.1 Proceso de obtención de los parámetros modales de la estructura

Las características dinámicas de puentes están definidas por sus frecuencias, formas modales y amortiguamiento en oscilaciones naturales o libres. Estos pueden estar resueltos por la excitación del puente, medición de la respuesta y análisis de datos. La determinación de frecuencia natural es importante desde su relación con la frecuencia contenida en función del forzamiento, teniendo mayor influencia en la respuesta del puente (Supriya and Kalyanshetti, 2013).

Para un mejor análisis de este proceso para nuestra investigación se propone el Diagrama 1.2, el cual demuestra la secuencia de pasos a seguir para llevar a cabo la obtención de los parámetros modales de la estructura, desde el proceso de obtención de los valores de las mediciones hasta la validación.

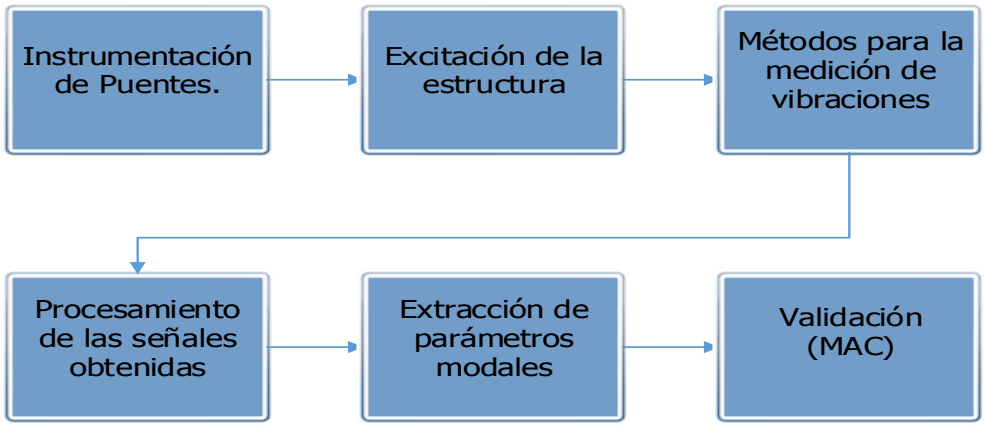


Diagrama 1.2: *Obtención de los parámetros modales de la estructura. Fuente: Elaboración Propia.*

En el caso de la instrumentación de puentes, esta permite obtener los resultados de las mediciones realizadas en los puntos requeridos, consiste en la colocación en puntos prefijados en el proyecto de prueba de carga aparatos de medida de flechas, deformaciones, giros, aceleraciones, etc. (Afonso, 2016, Rodriguez, 2010). Los instrumentos utilizados en este proceso pueden ser analizados y estudiados en (Afonso, 2016, Carrión Viramontes et al., 1999, Losa Miranda, 2015).

Los métodos más usados para la medición de vibraciones en puentes en la actualidad son el EMA (Análisis Modal Experimental) y el OMA (Análisis Modal Operacional) como una alternativa del EMA, los cuales arrojan los resultados necesarios para realizar el proceso de *Model Updating* (Soria et al., 2016, Losa Miranda, 2015, Claro, 2015, Anaya and Barajas, 2011). Otros métodos se pueden encontrar en (Reynders and De Roeck, 2014).

La extracción de los parámetros modales se realiza mediante los métodos en el dominio de la frecuencia (FRF) y/o los métodos en el dominio del tiempo (IRF) y la validación se concreta en una serie de herramientas para evaluar la calidad del modelo modal estimado (Anónimo, 2015).

El análisis modal experimental posee un último paso: la validación, que se concreta en una serie de herramientas para evaluar la calidad del modelo modal estimado o fases en el proceso de validación (Anónimo, 2015).

La finalidad de la fase de validación es verificar los resultados del paso anterior (extracción de los parámetros modales).

1.4 Obtención de parámetros estructurales del modelo

Este proceso tiene como objetivo obtener los parámetros modales reales del puente estudiado, determinar la respuesta real de la estructura en fuerzas o desplazamientos y posteriormente la identificación de los demás parámetros estructurales.

La obtención de los parámetros estructurales del modelo conlleva el uso de softwares y desarrollar los algoritmos correspondientes para la optimización de la estructura, donde se puede incluir en este caso la matriz Jacobiana en el algoritmo de calibración para mejorar los resultados en la minimización de las diferencias entre la respuesta medida y la modelada. Este proceso permite que sea posible utilizar el cambio de los parámetros dinámicos para determinar el daño en una estructura (Molina et al., 2012).

En el siguiente Diagrama 1.3 se hace referencia a dicho proceso, resaltando en este caso donde aparece incluida la matriz Jacobiana en el proceso de calibración del modelo del puente analizado.

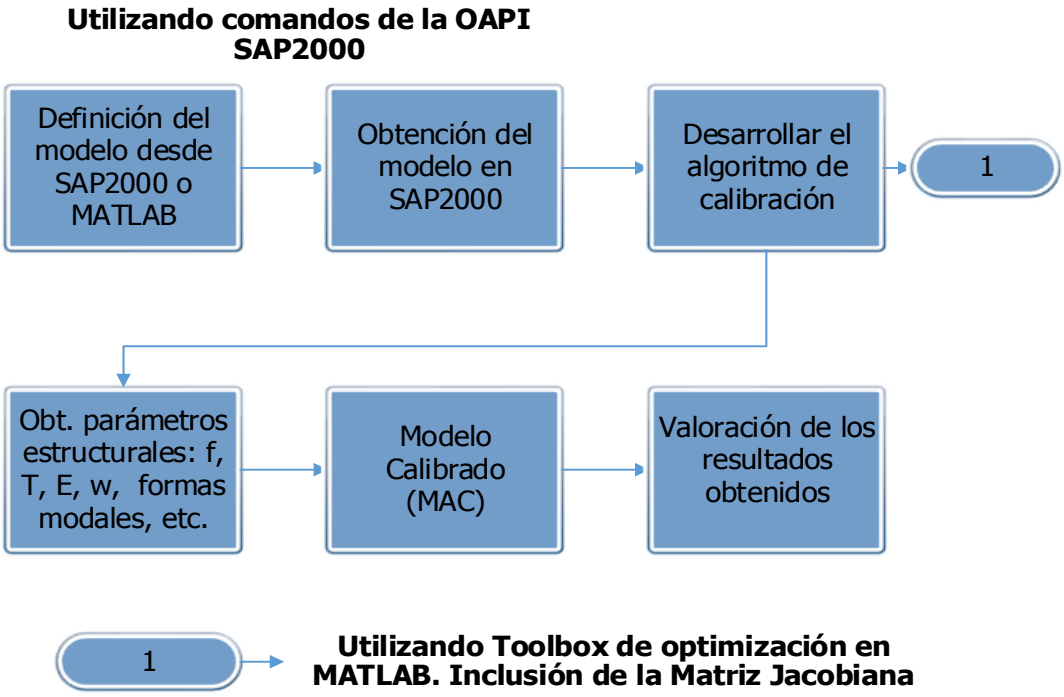


Diagrama 1.3: *Obtención de los parámetros estructurales del modelo. Fuente: Elaboración Propia.*

En este caso se llevan a cabo la confección del algoritmo de calibración llevado a cabo desde MATLAB para trabajar con el modelo del puente confeccionado en SAP2000 incorporando la matriz Jacobiana y obtener los parámetros estructurales. Se valoran los resultados teniendo en cuenta la rapidez del proceso con el uso o no de la matriz Jacobiana en la obtención del modelo calibrado. Una novedad aquí es el uso de la matriz MAC para el análisis de sensibilidad.

1.4.1 Modelación de Puentes

Modelar estructuralmente significa idear una estructura por medio de un modelo teórico factible de ser analizado con los procedimientos de cálculo disponibles (Afonso, 2016).

Para obtener el Modelo Mecánico o Modelo Analítico de la estructura real, se recomienda hacer un análisis integrado de las siguientes invariantes:

1- Modelación de la forma o geometría
2- Modelo del comportamiento de los materiales
3- Modelo de las acciones impuestas
4- Modelo de los apoyos y enlaces
5- Modelación de la Masa

Dar solución a un problema real a través de la modelación es un proceso que transita por varias etapas, lo cual aparece esquematizado en el Diagrama 1.4:

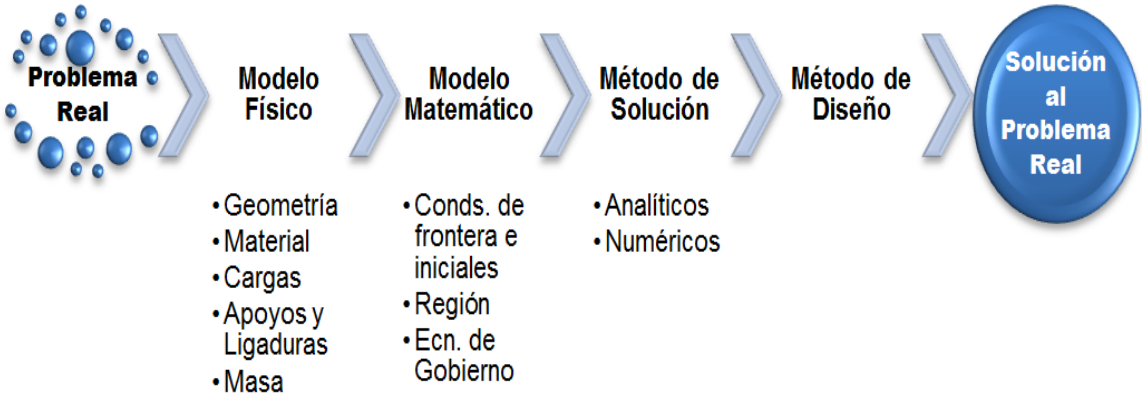


Diagrama 1.4: *Etapas de la modelación. Fuente: (Afonso, 2016)*

Una vez que se ha identificado el modelo y se ha programado, se necesita aplicarlo al problema concreto objeto de estudio. Para ello se necesita en primer lugar obtener los valores de los parámetros que utiliza el modelo. Estos pueden medirse directamente en el campo (sería la manera de proceder en un modelo físico) u obtenerse utilizando las técnicas de optimización que a partir de unos valores conocidos de las variables de entrada produzcan los correspondientes resultados en las variables de estado y de salida (sería la manera de proceder en los modelos empíricos) (Afonso, 2016).

Para realizar la modelación, análisis y diseño de puentes se han realizado investigaciones con numerosos softwares, dentro de estos están:

- ABAQUS
- ANSYS
- CSi Bridge
- SAP2000

La modelación en estos programas se hace empleando el MEF (Método de Elementos Finitos), que a su vez es un método de solución de problemas ingenieriles por aproximación, donde la región de integración que generalmente es continua. Se trabaja de forma que el elemento continuo se divide en un número finito de partes, cuyo comportamiento se especifica mediante un número finito de parámetros asociados a ciertos puntos característicos denominados nodos (Ramírez et al., 2015).

El Método de Elementos Finitos (MFE) forma parte de las técnicas de detección de daño. En caso de identificación de daño, el daño estructural es representado por una disminución en la rigidez de los elementos individuales y el procedimiento es realizado en dos procesos.

En el primer proceso, el modelo inicial de elementos finitos (FE) es sintonizado para la estructura intacta, lo cual es utilizado como un modelo remisivo. En el segundo proceso, el modelo remisivo FE está actualizado para obtener un modelo que puede reproducir los datos modales experimentales de la condición dañada. Los factores de corrección del proceso más reciente representan el daño. Para reducir el número de parámetros incógnitos, las funciones de daño se usan para aproximar el patrón desconocido de daño (Teughels, 2003).

1.4.2 Empleo de software para determinar parámetros estructurales en puentes (SAP2000-MATLAB)

Para el caso de esta investigación se hace referencia al estudio e implementación de los softwares SAP2000 conectado mediante la interface que posee (OAPI) con MATLAB el cual incluye en su paquete algunas de las técnicas más avanzadas de optimización que se utilizan en las estructuras, permitiendo mejorar considerablemente a la hora de diseñar y obtener estructuras óptimas.

1.4.2.1 MATLAB

MATLAB es el nombre abreviado de *MATrix LABoratory*. Es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares −tanto reales como complejos−, con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene también un lenguaje de programación propio (García de Jalón et al., 2005).

MATLAB es un programa de cálculo técnico y científico. Para ciertas operaciones es muy rápido, cuando puede ejecutar sus funciones en código nativo con los tamaños más adecuados para aprovechar sus capacidades de vectorización. MATLAB dispone de un código básico y de varias librerías especializadas (*Toolboxes*) (García de Jalón et al., 2005).

Es un sistema interactivo cuyo elemento básico de datos es un conjunto imponente que no requiere dimensionar. Permite solucionar muchos problemas técnicos de computación, ejemplos de los cuales incluya (Jaber, 2007):

- Trabajos con matrices
- Encontrando las raíces de polinomios
- Las aplicaciónes digitales (la caja de herramientas) de procesamiento de la señal
- La esquematización: Los de x-y y gráficos polares 3D

Particularmente útil para:

- El desarrollo de algoritmos
- El modelado y la simulación
- La adquisición de datos
- El análisis de datos, la exploración, y la visualización
- El desarrollo de aplicaciónes, incluyendo edificio gráfica de la interfaz del usuario.

Toolboxes más importantes (Bacuilima, 2011):

- *Curve fitting*: Ajustes de modelos y análisis.
- *Data Acquisition*: Adquiere y envía datos a un instrumento electrónico conectado al computador. (sólo para Windows)
- *Excel link*: Permite usar MATLAB con datos leídos directamente desde planillas Excel.
- *Image processing*: Realiza el procesamiento de imágenes, análisis y desarrollo de algoritmos.
- MACEC: Es un *toolbox* programado sobre MATLAB para el análisis modal operacional y experimental.
- *Partial differential equation*: Soluciona y analiza sistema de ecuaciones diferenciales parciales.
- *Signal Processing*: Posibilita el procesamiento de señales, análisis y desarrollo de algoritmos.
- *Spline*: Crea y manipula modelos de aproximación de datos Spline.
- StaBIL-2.0: *Toolbox* para el análisis por MEF de estructuras (Dooms et al., 2010).
- *Statistics*: Aplica modelos estadísticos y modelos de probabilidades.
- *Structural Dynamics*: Analiza modelos de elementos finitos y lleva a cabo análisis modales de sistemas mecánicos.
- *Wavelet*: Analiza, comprime y saca el ruido de señales e imágenes usando técnicas de wavelet.
- *Optimization*: Consta de un conjunto de funciones que resuelven problemas de extremos, con o sin condiciones, de funciones reales las cuales son generalmente multivariables y no lineales.

Actualmente en el mundo industrial, este software (MATLAB) es utilizado como una de las principales herramientas de investigación para la resolución de complejos problemas planteados en la realización y aplicación de modelos matemáticos en ingeniería.

Los usos más característicos de esta herramienta se encuentran en áreas de computación y cálculo numérico tradicional, prototipos algorítmicos, teoría de control automático,

estadística, análisis de series temporales para el proceso digital de señales (Bacuilima, 2011).

1.4.2.2 *Toolbox* de Optimización

El *toolbox* de optimización con el que consta MATLAB posee un conjunto de funciones que resuelven problemas de extremos, con o sin condiciones, de funciones reales las cuales son generalmente multivariables y no lineales. Del mismo modo, posee funciones para la resolución de algunos tipos de problemas matriciales en extremos. Resulta conveniente para una comprensión y mejor manejo del *toolbox*, poseer conocimientos básicos previos de análisis de funciones reales, matrices y teoría de extremos (Bacuilima, 2011).

Entre las principales funciones de la caja de herramientas de optimización están (Bacuilima, 2011):

- *fminbnd*: Esta función resuelve los problemas de optimización de funciones de una variable sin restricciones, se la conoce también como optimización escalar.
- *fminsearch*: Esa función en cambio resuelve los problemas de optimización de funciones de más de una variable sin restricciones.
- *fminunc*: Esta al igual que la anterior proporciona el mínimo de una función de variables sin restricciones, pero, con la diferencia que utiliza información del gradiente y el hessiano de la función objetivo.
- *fmincon*: Esta función determina el mínimo de una función multivariable con restricciones de igualdad y desigualdad, lineales y no lineales.
- *quadprog*: Realiza la minimización de una función cuadrática con restricciones de igualdad y desigualdad lineales.
- *linprog*: Esta función realiza la optimización de problemas de programación lineal.
- *lsqnonlin*: Resuelve por mínimos cuadrados problemas no lineales de funciones o de ajuste de datos.
- *ga-Genetic Algorithm*: Resuelve por Algoritmos Genéticos diferentes tipos de problemas con o sin restricciones (Afonso, 2016).

Al analizar cada una de estas funciones y su aplicación, la función que más se ajusta para resolver el problema de *Model Updating* es la *lsqnonlin*, pues a través de la misma se realiza la optimización de problemas no lineales por mínimos cuadrados mediante el ajuste de datos o valores obtenidos del software SAP2000 mediante la interface OAPI que se realiza con MATLAB (Reynders and De Roeck, 2014).

En este caso también MATLAB permite la creación de interfaces con otras plataformas mediante las cuales se puede llevar a cabo una resolución conjunta del problema de calibración de un modelo estructural mediante técnicas de optimización, siendo más específico en (González, 2015). El empleo de la OAPI SAP2000-MATLAB, puesta en práctica anteriormente por las investigaciones recientes de (Bacuilima, 2011, Sextos and Balafas, 2011, Afonso, 2016) enriquece esta investigación.

1.4.2.3 SAP2000

El SAP2000 es un programa de amplio espectro que puede servir desde los problemas más simples o los proyectos más complicados. El SAP2000 es un programa basado en EF (Elementos Finitos), con interfaz gráfico 3D orientado a objetos, preparado para realizar, de forma totalmente integrada, la modelación, análisis y dimensionamiento de los más amplios conjuntos de problemas de ingeniería de estructuras.

Conocido por la flexibilidad en el tipo de estructuras que permite analizar, por su poder de cálculo y por la fiabilidad de los resultados, SAP2000 es la herramienta de trabajo diario para varios ingenieros. La versatilidad en modelar estructuras, reconoce su utilización en el dimensionamiento de puentes, edificios, estadios, presas, estructuras industriales, estructuras marítimas y todo tipo de infraestructura que necesite ser analizada y dimensionada (Computers and Structures, 2013).

SAP2000 está basado en objetos, queriendo decir que los modelos son creados usando miembros que representan la realidad física. Los resultados del análisis y el diseño son informados para el objeto en conjunto, y no para cada sub-elemento que conforma el objeto, suministrando información que es tanto más fácil interpretar como más consecuente con la estructura física (Computers and Structures, 2013).

Las diversas herramientas de análisis y los procesos desarrollados en SAP2000 permiten la evaluación de grandes desplazamientos en cada etapa de la estructura, el análisis modal a través de los vectores propios *Eigen* y *Ritz* basados en casos de carga no lineales, el análisis del comportamiento catenaria en cables, la no linealidad del material *(rótulas fiber)* y de los objetos de área no lineales (*layered shell*), el análisis de pandeo o colapso progresivo, el uso de «*links*» no lineales para modelado de la fricción, amortiguación, aisladores de base y rigidez multilineal o plástica entre nudos, y finalmente, la secuencia constructiva. Los análisis no lineales pueden ser estáticos y/o en función del tiempo, con opciones para análisis dinámico FNA (*Fast Nonlinear Analysis*), temporales (*time-history*) y por integración directa (Computers and Structures, 2013).

Desde sencillos modelos estáticos utilizados para análisis 2D, a los modelos complejos y de grandes dimensiones que requieren análisis avanzados no lineales, el programa SAP2000 es la solución más eficiente y productiva para los ingenieros de estructuras en general.

1.4.2.4 OAPI SAP2000

CSi Open Application Programming Interface (OAPI) es una potente herramienta que permite a los usuarios automatizar muchos de los procesos requeridos para construir, analizar y diseñar modelos y obtener resultados del análisis y diseño personalizados y enlazar SAP2000 con otro software, proveyendo una vía para un intercambio de la información del modelo con otros programas (Chagoyén, 2015).

Es una poderosa herramienta que le cede al usuario la apropiación de numerosos procedimientos requeridos para construir, analizar y diseñar modelos y obtener resultados optimizados de los mismos. Además, consiente al usuario en la creación de vínculos entre SAP 2000 y softwares de última generación, suministrando una guía para el intercambio de dos flujos de información del modelo con estos programas (Afonso, 2016).

En cuanto a la programación informática, consiste en una biblioteca de software que ofrece acceso a una colección de funciones capaces de controlar a «distancia» la forma en que se comporta el SAP2000, por lo tanto, anula el procedimiento de hacer clic sobre el SAP2000 para poder trabajar en él. Los aspectos más relevantes de las principales características que ofrece la OAPI se resumen a continuación (Sextos and Balafas, 2011):

- Directo, acoplamiento rápido y sólido con los métodos de diseño y análisis del SAP2000.
- El flujo de datos es de dos vías, ya que puede ser utilizado para facilitar ambos procedimientos pre y post-procesamiento.
- No hay necesidad de usar archivos intermedios, lo que reduce significativamente el tiempo necesario para el intercambio de datos cuando se trabaja en modelos de gran tamaño.
- Compatibilidad con la mayoría de los lenguajes de programación más importantes.
- Transferencia de datos y el control simultáneo de un modelo estructural por diferentes aplicaciónes de terceros.
- Desarrollo de aplicaciónes de terceros que permanecerán compatible con futuras versiones de SAP2000.

- Capacidad para desarrollar una interfaz personalizada para SAP2000, calibrado para las necesidades del usuario o para incorporarla en una aplicación que permite la programación del usuario.

1.4.2.5 Uso de la OAPI SAP2000

Para encontrar la ubicación del mismo vale resaltar que se proporciona además en el programa junto con la instalación un archivo con la documentación detallada *(CSi_OAPI_Documentation.chm)* como asistente general en la utilización de la OAPI de SAP2000, que incluye toda la información necesaria que ayudará a acostumbrarse a la programación de la OAPI.

Esta guía es para ser utilizada en el desarrollo del software SAP2000, ya que contiene una lista completa de todas las funciones únicas previstas, acompañadas de su sintaxis exacta, una descripción detallada de los argumentos que maneja y un ejemplo comentado de su uso (Sextos and Balafas, 2011).

Para introducirnos en este medio se requiere una base de programación sólida, ya que el programa contiene una amplia gama de lenguajes de programación que se puede utilizar que cubren la gran mayoría de las modernas opciones de desarrollo de software, incluyendo Visual Basíc.NET, Visual Basic para Aplicaciones (VBA), Visual C #, Visual C ++, visual Fortran y MATLAB (Ghedan and Nildem, 2013).

La OAPI posee un grupo de funciones previstas que realizan la misma función que el propio SAP2000 en su ventana de trabajo. Estas permiten la interacción del SAP2000 con un segundo software dependiendo del lenguaje de programación que se utilice. Estas funciones están resumidas en (Chagoyén, 2015, Afonso, 2016).

1.4.3 Algoritmo para calibrar el modelo de elementos finitos

Se necesita un algoritmo para lograr la interface OAPI-SAP2000, lo cual posibilita realizar la modelación y la calibración de los modelos realizados en SAP2000 a partir de la versión 11 o desde MATLAB con el correspondiente lenguaje de programación. El algoritmo desarrollado puede simplificarse según lo ilustrado en el Diagrama 1.5:

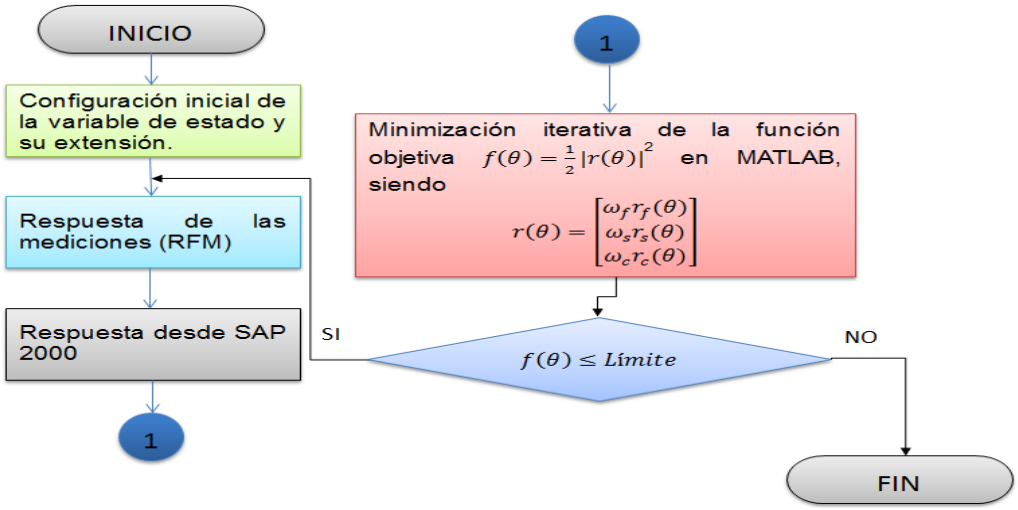


Diagrama 1.5: *Algoritmo para calibrar el modelo. Fuente:(Afonso, 2016)*

Primero se necesita un estudio extensivo de la teoría matemática de optimización para llevar a cabo los algoritmos de actualización. Esta teoría es críticamente sintetizada y claramente ilustrada con ejemplos numéricos en (Teughels, 2003).

Para la elaboración del algoritmo de actualización del modelo de FE, se tiene en cuenta el procedimiento básico y las mejoras que puede tener el mismo usando las estrategias más efectivas conocidas de la disciplina de optimización. En este caso los temas específicos en el problema de la formulación de dicho algoritmo tanto como para la definición del mismo, conlleva, la selección de parámetros, trabajo con la función objetivo, el cálculo de la matriz de sensibilidad o matriz Jacobiana, el algoritmo de optimización específico que posee el software con que se trabaja y la medida de condicionamiento del problema (Teughels, 2003).

Aunque una función mínima de cuadrados puede ser minimizada con una técnica general espontánea de minimización, los algoritmos específicos para los problemas de mínimos cuadrados están disponibles en MATLAB mejoran la eficiencia iterativa del procedimiento de solución. Aquí el Jacobiano puede ser introducido en el algoritmo por el propio usuario o aproximado con cálculos de diferencias finitas (Teughels, 2003).

En este caso de estimación de parámetros modales, para calibrar el modelo se emplea un algoritmo de optimización basado en el método de la región de confianza *Gauss-Newton* (Reynders et al., 2007), implementando en MATLAB la función: *lsqnonlin* (Reynders and De Roeck, 2014).

El vector θ contiene los parámetros de rigidez a actualizar. Suponiendo que el vector $r(\theta)$ contiene las diferencias entre los parámetros del modelo numérico con las propiedades de la rigidez y los parámetros modales medidos. La forma común de resolver el problema de calibración es mediante la minimización del vector residual $r(\theta)$ empleando mínimos cuadrados, lo cual corresponde a la minimización de la función objetivo (Reynders et al., 2007).

1.5 Conclusiones parciales del capítulo

Luego de realizado el marco teórico referente al estado del conocimiento sobre calibración de modelos estructurales se arriban a las siguientes conclusiones.

- La actualización de un modelo FE es un procedimiento que puede considerarse como un proceso de minimización en el cual las discrepancias entre los datos numéricos, experimentales y modales son minimizados, ajustando los parámetros desconocidos del modelo.
- La asociación entre MATLAB y SAP2000, programas de análisis estructural y programación respectivamente, mediante la OAPI, facilita además de la modelación, análisis y diseño, desarrollar el procedimiento de calibración de forma automatizada, logrando obtener resultados satisfactorios.
- Los algoritmos de calibración utilizados pueden ser modificados con el objetivo de aumentar la velocidad de convergencia de este, por lo cual se recomienda incorporar expresiones analíticas de la matriz Jacobiana.
- La matriz Jacobiana está formada por las derivadas parciales de primer orden de la función objetivo que se va a minimizar, esta va buscando los valores donde el gradiente de la función objetivo cambia su signo lo cual indica que el punto óptimo está próximo al analizado.
- El objetivo de realizar un análisis de sensibilidad es identificar cuáles de estos factores tienen una mayor influencia y modifican el comportamiento de la estructura. El uso de la matriz MAC admite la validación del modelo y su calibración logrando una correlación perfecta entre las formas modales del modelo numérico y las formas modales de la estructura real.
- El proceso de calibración de parámetros modales tiene como objetivo encontrar los valores óptimos para estas variables desconocidas de tal manera que el error entre estos parámetros modales, analíticos y experimentales, se minimice y se realice a través de una ecuación llamada función objetivo o función error que relaciona las frecuencias y formas modales de vibración.

Capítulo II: Procedimiento para la calibración de modelos estructurales utilizando la interface OAPI SAP2000-MATLAB incluida la matriz Jacobiana

En este capítulo se realiza el procedimiento para la calibración de modelos estructurales, específicamente, modelos de puentes, lo que tiene como objetivo la elaboración del algoritmo de interface entre MATLAB y SAP2000 que incluya la matriz Jacobiana para la obtención del modelo calibrado. Todo esto puede ser desarrollado desde MATLAB para crear el modelo estructural del puente objeto de estudio o interactuar con el modelo ya creado en SAP2000. El fin es buscar los parámetros de la respuesta dinámica modelada y realizar la elección e implementación de la función objetivo y el tipo de optimización oportunos para ser aplicados a este problema.

2.1 Introducción

La calibración de modelos mecánicos de puentes es un paso importante y necesario con fines de validar dichos modelos, el mismo se basa en la elaboración e implementación de un algoritmo para realizar la actualización. La obtención del modelo calibrado permite estimar los parámetros estructurales del puente analizado, lo que proporciona un mejor análisis en cuanto a su comportamiento estructural ante situaciones de carga o estado de la estructura que no pueda ser experimentalmente ensayado, evaluación de cargas, detección de desperfectos y daños estructurales.

El procedimiento para obtener la actualización del modelo conlleva en sus inicios la recopilación de información de la estructura del puente, la realización de los ensayos modales y la modelación de la estructura, para así obtener el modelo calibrado, lo que equivale a la obtención de los parámetros estructurales deseados. Se lleva a cabo un proceso que transita por diversas etapas y en las que cada una juega un papel sumamente importante. El mismo empieza con los trabajos de campo y termina con trabajos de oficina.

En este caso el procedimiento empleado se lleva a cabo mediante la programación y optimización en el software MATLAB R2015a y la modelación de la estructura en SAP2000v16, incluyendo en el proceso la matriz Jacobiana *(off-on),* logrando el intercambio de información entre ambos a través de la OAPI SAP2000.

2.2 Generalidades del método utilizado en la optimización

Para llevar a cabo un análisis profundo de cómo se aclara el proceso de optimización con sus algoritmos hay que tener en cuenta la base matemática de los métodos utilizados por varios autores (Simoen, 2013, Teughels, 2003) los cuales están desarrollados desde los más simples hasta los más complejos.

En esta investigación como se explicó anteriormente el método de optimización utilizado es el de Optimización Local ya que los algoritmos que convergen más rápido resultan solo en una solución local. Este trabajo también hace uso del estudio de sensibilidades, la cual ha demostrado que en su utilización concibe que se obtengan resultados satisfactorios y una rápida convergencia.

Algunos rasgos de los Métodos de Optimización que utilizan el estudio de sensibilidades se presenta a continuación (Maes et al., 2017):

Los métodos locales de optimización: (*Newton, Quasi-Newton*)

- Rápida convergencia (1 punto de búsqueda, basado en sensibilidades).
- El resultado depende del punto de inicio, no garantizando el mínimo global.

Los métodos globales de optimización:

a) Con búsqueda aleatoria (Algoritmos Genéticos, *Simulated Annealing SA*.)

- Se obtiene el mínimo global, el resultado es independiente del punto(s) de inicio.
- Se evalúa un enorme número de funciones, desaceleran la convergencia.

b) Basado en sensibilidades (CLM)

- Se obtiene el mínimo global, el resultado es independiente del punto(s) de inicio.
- Rápida convergencia (basado en sensibilidades).

Los métodos para resolver estos problemas son iterativos y cada paso de iteración usualmente requiere la solución de un problema relacionado con optimización de mínimos cuadrados. En esta investigación el problema mínimo no lineal de cuadrados es solucionado con el método basado en sensibilidades-iterativo *Gauss-Newton*. Esto quiere decir que la matriz Hessiana es aproximada usando sólo el término de primer orden con el uso de la matriz Jacobiana (Teughels, 2003).

El método de *Levenberg-Marquardt* también hace querencia de este procedimiento, lo cual permite también el desarrollo de la programación propuesta en esta investigación.

El método *Gauss-Newton* es un método de búsqueda lineal y excluye el término de segundo orden. Se distingue por presentar la dirección y la magnitud del cambio en las

variables del diseño en cada iteración. Los cambios de la parte pequeña en la dirección del *Gauss-Newton* disminuirán la suma de cuadrados, pero el tamaño del cambio indicado por el método puede ser tan grande que en lo sucesivo itere-oscile, y en el peor caso, el procedimiento se haga inestable. Este puede lograr un rango de convergencia cuadrático, a pesar de que solo las primeras derivadas son utilizadas.

Además de la medida de disminuir el número de variables del diseño usando la función de daño, el método del *Gauss-Newton* es modificado y mejorado implementándolo con un algoritmo de la región de confianza. La región de confianza impide la iteración de pasos muy grandes y consecuentemente evita la divergencia del proceso. Su radio está actualizado en cada iteración según la exactitud de la función de modelo que aproxima. La estrategia tiende a reducir oscilaciones en las variables del diseño y por lo tanto hace el método de optimización más fuerte (Teughels, 2003).

Conjuntamente a la implementación de la región de confianza, también las restricciones destinadas explícitas se introducen en el problema de minimización para sobrecoger la inestabilidad de llegar muy lejos en el método del *Gauss-Newton*. Las restricciones destinadas tienen un efecto favorable en la estabilidad del problema en el sentido que reduce el espacio de búsqueda y consecuentemente evita divergencia. Debe ser especificado que los límites físicos no pueden ser excedidos por las variables.

El algoritmo de *Levenberg-Marquardt* puede ser visto como una modificación con región de confianza al método de *Gauss-Newton*. El uso de una región de confianza evade la principal debilidad del método de *Gauss-Newton* en su comportamiento cuando el Jacobiano presenta un rango deficiente. En este método el término de segundo orden de la Hessiana es todavía ignorado, por lo que la convergencia de este método es muy similar al de *Gauss-Newton*. Para problemas con residuos pequeños o casos casi lineales se logran rápidas convergencias; pero si los residuos son grandes o los problemas son no lineales, este método tendrá convergencias lentas. También existen otros métodos más robustos y complejos para estos casos (Teughels, 2003, Simoen, 2013).

2.3 Procedimiento para la modelación de puentes empleando SAP2000v16

2.3.1 Recopilación de información sobre el puente objeto de estudio

Dentro de los pasos y aspectos para la recopilación de información sobre el puente objeto de estudio se encuentran los trabajos de campo, en los que se realizan distintos tipos de ensayos lo cual es de vital importancia ya que permiten recopilar toda una serie de información de vital interés para construir el modelo del puente lo más acercado a la

realidad, lo que trae consigo un mejor análisis estructural (González, 2015). Como resumen de dicha metodología planteada en investigaciones anteriores se han dividido en dos grupos, siendo reflejadas en el Diagrama 1.6:

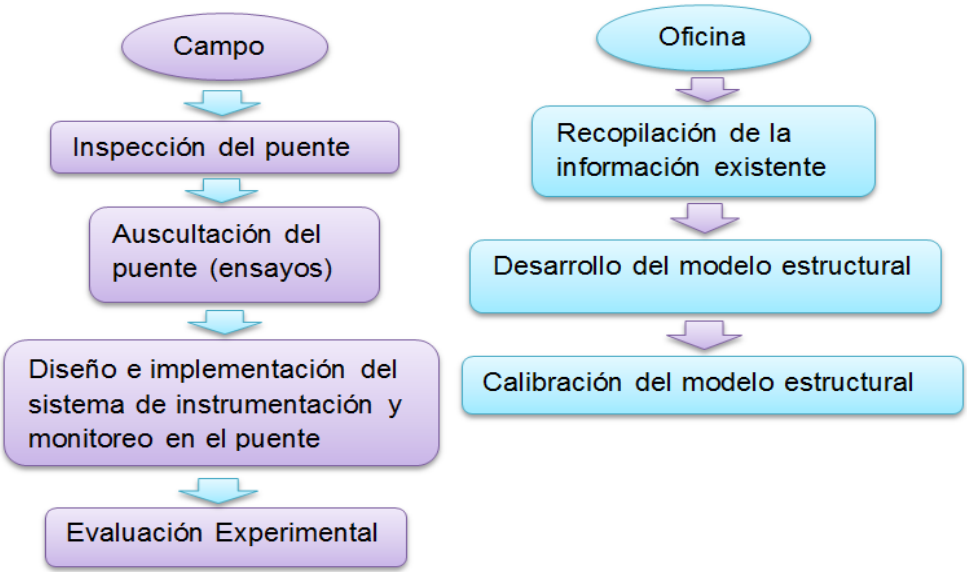


Diagrama 1.6: *Etapas que componen la metodología propuesta. Fuente:(González, 2015)*

Este procedimiento puede ser ayudado por estudios de tráfico, planos de diseño que contengan detalles constructivos, de mantenimiento y rehabilitación confeccionados desde la etapa de diseño del puente antes de la puesta en ejecución de la obra. Algunos datos de interés pueden ser como parte de la inspección de puentes: levantamiento de las características fundamentales y patologías; de la auscultación de puentes: propiedades físicas, químicas y mecánicas de los materiales, comportamiento estructural o evolución de defectos constatados; del diseño e implementación del sistema de instrumentación y monitoreo en el puente: las señales obtenidas en el equipo de adquisición de datos, el procesamiento de estas señales y obtención de los parámetros modales de la estructura y de la evaluación experimental: los resultados de ensayos de diagnóstico, de pruebas estáticas y dinámicas, de pruebas de carga, etc. Con estos datos se puede comenzar a modelar la estructura, demostrando que es de vital importancia poseer la mayor cantidad de información posible.

2.3.2 Generación del modelo estructural del puente en SAP2000v16

El modelo estructural del puente realizado en SAP2000v16 es creado cumpliendo en secuencia lógica con cada una de las invariantes de la modelación y adecuado para que posteriormente al ser reconocido y analizado a través de la interface con MATLAB R2015a

no surjan incertidumbres en cuanto a los valores obtenidos. Los pasos a seguir son explicados de forma similar en (Afonso, 2016)

Los pasos a seguir son:

1. **Abrir el SAP 2000v16.**
2. **Definición de unidades:** En SAP2000 siempre se definen primero las unidades de medida con las que se van a trabajar antes de elaborar el modelo mecánico del puente, aunque posteriormente en el proceso de creación del mismo estas se puedan cambiar. Desde MATLAB se pueden cambiar empleando el comando (*SetPresentUnits*) de la OAPI.
3. **Geometría del modelo estructural:** Utilizando todas las herramientas de dibujo (*Draw*) con que cuenta el SAP2000 se crea el modelo del sistema estructural del puente lo más cercano posible a la realidad, además empleando las herramientas de definición (*Define*) se especifican las secciones de los diferentes elementos estructurales que componen el modelo.
4. **Definición de los materiales:** Con la herramienta (*Define*) se asígnan las propiedades mecánicas de los materiales (obtenidas de los ensayos realizados al puente). Luego de definidas las secciones de los elementos que componen el modelo se especifica el material correspondiente a cada uno de ellos. Aunque para las propiedades que serán objeto de estimación (variables independientes en el algoritmo de optimización) se puede asignar cualquier valor, se recomienda utilizar los valores resultantes de los ensayos o valores lógicos si no se disponen de estos.
5. **Definición de los apoyos:** En este caso la herramienta de asígnación del SAP2000 (*Assign*) permite modelar múltiples condiciones de apoyo que deben estar en correspondencia con la realidad. Este tema de los apoyos puede ser considerado como una variable independiente más en el problema de optimización.
6. **Modelación de las cargas:** Las cargas a definir en el modelo deben ser las que estaban actuando sobre la estructura en el momento que se instrumentó, para que así el modelo represente lo que se va analizar. En este caso solo se asignarán las cargas muertas impuestas por el peso propio de los elementos sobre el puente y de los elementos estructurales que lo conforman. Aquí hay que destacar que solamente se involucra a las cargas para asignar las masas posteriormente debido a que el caso de carga a analizar es el Modal impuesto por defecto en SAP2000 en la ventana de *Define Load Cases*. Esto se debe a que los valores que se van a tomar como resultado del análisis en el SAP2000 van a ser las coordenadas modales y las frecuencias propias de los diferentes modos de oscilación analizados.

7. **Modelación de las masas:** En SAP2000 existen diferentes formas de definir las masas. Estas están disponibles en el programa en las propiedades de los materiales (*Material Properties*), en las cargas verticales aplicadas (*Applied Vertical Loads*) o en asígnación de masas a objetos (*Assigned Object Masses*). Para el primer caso se emplea la herramienta *Define* y en las propiedades de los materiales se coloca el peso por unidad de volumen del material, el programa calcula la masa por unidad de volumen dividiendo entre la gravedad. El segundo caso se emplea a través de la herramienta *Assign*, dentro de esta en *Joint Loads* en *Forces* para las cargas en la dirección z negativa, aquí la carga es dividida igualmente por la gravedad. El tercer caso se ejecuta también desde la herramienta *Assign*, se selecciona dentro de *Joint* el que dice *Masses*. Por otra parte, para especificarle al programa donde encontrar la masa se busca en la herramienta *Define* el comando *Define Mass Source*, después se elige si la masa proviene de los elementos y masas adicionales o si es de casos de carga o de ambas partes.
8. **Definición del caso de carga:** Para este caso en la herramienta *Define* se selecciona *Load Case* y dentro de este el *Modal* en el que se puede especificar el tipo de análisis: *Modal-Dead*, además del número máximo y mínimo de modos de vibración, el tipo de vectores a utilizar, etc.
9. **Definición del tipo de análisis y corrida del modelo:** El análisis a ejecutar es el *Modal* por lo que en la herramienta *Analyze* se selecciona en *Set Analysis Options* el comportamiento que tendrá (*Space Frame, Plane Frame*). Luego se corre el análisis en *Run Analysis* poniendo solo a correr el caso *Modal.* Para el caso de la obtención de las matrices de masa y rigidez dentro de la propia opción de *Analyze* se selecciona *Set Analysis Options* y en la ventana que aparece, marcar *Solver Options,* dando paso a otra ventana con el nombre *Equation Solver Options* en la cual se selecciona *Advanced Solver*, *Auto* en *Analysis Process Options* y en el *Analysis Case Name: Modal/Dead*. Luego ejecutamos el modelo normalmente y se obtienen los resultados. Dentro del directorio donde se encuentra nuestro modelo obtendríamos la salida de ficheros con extensión *.TXA, *.TXE, *.TXM, *.TXK, *.TXC donde:
   a) ***.TXA:** contiene la explicación del contenido de los demás ficheros.
   b) ***.TXE:** indica el número de la ecuación para cada DOF en cada nodo.
   c) ***.TXM:** almacena la matriz triangular inferior simétrica de masas.
   d) ***.TXK:** almacena la matriz triangular inferior simétrica de rigidez.
   e) ***.TXC:** contiene los *constrains* de SAP2000 y los incorpora a K.
10. **Guardar fichero del modelo *.sdb:** Luego de la corrida del fichero se debe guardar este archivo *.sdb en la carpeta de la función que se creará en MATLAB para interactuar con el mismo, pues esta carpeta será añadida al directorio de MATLAB. El nombre

colocado debe corresponderse a la ruta definida en esta función para que los comandos de la OAPI declarados abran y operen sobre el modelo.

2.4 Procedimiento implementado en MATLAB para la optimización

2.4.1 Matriz Jacobiana en MATLAB (*off-on*)

Sintaxis General: Jacobian (f, v)

Siendo f: Escalar o función vectorial. Si la f es un escalar, entonces la matriz Jacobian de f es la gradiente invertida de f.

Para el caso de v: Vector de variables con relación a las cuales usted computa Jacobian. Si la v es un objeto simbólico vacío, algo semejante como *sym* ([ ]), entonces el Jacobian devuelve un objeto simbólico vacío.

Puede tener muchos usos de los cuales se encuentran los siguientes:

- Matriz Jacobiana de una Función Vectorial: es una matriz de las derivadas parciales de esa función.
- Matriz Jacobiana de una Función Escalar: es la inversa de su gradiente.
- Matriz Jacobiana con relación a un Escalar: es la primera derivada de esa función. Para una función vectorial, el Jacobiano con relación a un escalar es un vector de las primeras derivadas.

La matriz Jacobiana posee muchos usos dentro del espectro de programación de MATLAB, la misma se utiliza en cálculos de matrices desde los más simples a los más complejos dentro del *Symbolic Math Toolbox*, en ecuaciones no lineales, en el uso de los problemas de mínimos cuadrados no lineales y lineales dentro del *Optimization Toolbox* siendo en este caso su uso a preferencia del usuario; es decir, se utiliza en (*'Jacobian', 'off' / 'on'*) dependiendo de los resultados que se quieran obtener.

Para el caso de esta investigación la matriz Jacobiana se encuentra formando parte de la función de optimización a utilizar: *lsqnonlin*. En este caso se utiliza el Método de Nelson (*Nelson's Method*) para el cálculo de la matriz Jacobiana (Maes et al., 2017).

2.4.1.1 Método de Nelson (*Nelson's Method*)

El cálculo de las derivadas de las frecuencias naturales y las formas de modos con respecto a los parámetros del modelo es de vital importancia para la optimización del diseño, *Model Updating* en elementos finitos, detección de daño y otras aplicaciones. Muchos autores han trabajado con estas derivadas estableciendo diferentes formas para

su determinación (Fox and Kapoo, 1968, Nelson, 1976, Friswell and Adhikari, 1976, Adhiakri, 1999), formulando una amplia base matemática para el desarrollo del tema, la cual es modificada a través de los años.

El Método de Nelson introduce la aproximación, en este solo se requiere de los vectores propios de interés, lo que hace una ventaja para el uso del método. Los resultados se presentan en términos de los cambios en masa, amortiguamiento, los matrices de rigidez y las complejas soluciones propias de segundo orden a fin de que la representación del estado-espacio de las ecuaciones de movimiento pueda ser evitada.

El procedimiento simplificado para determinar derivadas de los vectores propios se hace más eficiente que otros procedimientos cuando se consideran grandes sistemas. La creación de algoritmos con este método debería hacer tales cálculos suficientemente exactos y económicos para permitir estudios eficientes de optimización de grandes sistemas de vectores propios de sensibilidad.

El Método de Nelson exige un acercamiento para computar las derivadas del vector propio que sólo requiere el trabajo de los datos modales para los N_m de los modos de interés y la solución de $N_d \; x \; N_d$ conjunto lineal de ecuaciones para cada conjunto de parámetros evaluados. Los pasos a tener en cuenta para el desarrollo de dicho algoritmo son los siguientes (Simoen, 2013):

1: Calcular la matriz $\mathbf{A_r}$ y el vector $\mathbf{b_{ri}}$:

$$Ar = K - \lambda r M \qquad \text{K: rigidez, M: masa} \qquad (5)$$

$$b_{ri} = -\left[\frac{\partial k}{\partial \theta Mi} - \frac{\partial \lambda}{\partial \theta Mi} M\right] \varphi r \qquad \partial \text{k: medir sensibilidad dK} \qquad (6)$$

2: El sistema de matrices $\mathbf{A_r^*}$ y el vector $\mathbf{b_{ri}^*}$ son obtenidos por modificar los originales matrices Ar y $\mathbf{b_{ri}}$:

$$A_r^*: [Ar]k = 0 \quad [Ar]k = 0 \quad [Ar]kk = 1 \qquad (5.1)$$

$$b_{ri}^* \; ; \; [bri]k = 0 \qquad (6.1)$$

Donde la k es un índice para el cual el vector propio es diferente a cero. Usualmente el valor absoluto máximo del vector propio es usado como: $[\varphi r]k = \max(\left|\, \varphi r \,\right|)$ (7)

3: Calcular los vectores propios de sensibilidad:

$$\frac{\partial k}{\partial \theta Mj} = (I - \varphi r \varphi_r^T M)(A_r^*)^{-1} b_{ri}^* \qquad \text{I: Matriz idéntica} \qquad (8)$$

Este método es programado dentro del script confeccionado en MATLAB para la calibración de los modelos.

2.4.2 Declaración y explicación de la función de optimización empleada en MATLAB: (Función *lsqnonlin*)

2.4.2.1 *lsqnonlin* en problemas de mínimos cuadrados

Para algoritmos de mediana escala los métodos de *Gauss-Newton* y *Levenberg-Marquardt* son suministrados por MATLAB. Los dos son implementados como métodos de búsqueda directa formando parte de la función *lsqnonlin* (Teughels, 2003). En el primero la dirección de búsqueda es calculada resolviendo:

$$\left(J_k^T J_k\right) p^{GN} = -J_k^T r_k \qquad (9)$$

En el segundo una dirección regularizada:

$$\left(J_k^T J_k + u_k I\right) p^{LM} = -J_k^T r_k \qquad (10)$$

Donde:

J_k – Matriz Jacobiana.

r_k – Residuos.

I – Matriz idéntica (1 en la diagonal principal, 0 fuera de la diagonal principal).

p^{GN} – Distancia del paso de Gauss-Newton.

p^{LM} – Distancia del paso de Levenberg-Marquardt.

u_k – Parámetro que limite el tamaño de p; si $u_k = 0$ entonces p^{GN}=p^{LM}.

Esta es utilizada para mejorar la condición del problema. La dirección de la búsqueda de *Gauss-Newton* es calculada exactamente con una descomposición QR (factores de factorización QR) de J_k (matriz de sensibilidad de z con respecto a θ de la interacción con k), sin formar la ecuación (9) explícitamente, requiriendo solo pocas iteraciones y tiempo de cálculo para problemas a medias escalas. El método de Levenberg-Marquardt requiere un poco más de cálculos a partir de que u_k tiene que ser controlada en cada iteración.

Subsecuentemente, una búsqueda en una dirección es realizada a través de la dirección de búsqueda con un algoritmo iterativo. El Jacobino puede ser suministrado por el usuario o aproximado con una diferencia finita de cálculos (Teughels, 2003).

Los algoritmos de gran escala son métodos basados en la estrategia de región de confianza. Análogamente, así como en la minimización general, una función modelo cuadrática es construida en cada iteración que aproxima la función objetivo. En la aplicación de mínimos cuadrados, la Hessiana es aproximada con la formulación de *Gauss-Newton*. Las derivadas de segundo orden son por consiguiente no usadas.

En particular, el algoritmo es implementado tal que la dirección de *Gauss-Newton* se usa para definir el subespacio de dos dimensiones. Esta dirección es la solución de:

$$\min_{p} \| J_k p + r_k \|^2 \qquad (11)$$

El método de los gradientes conjugados pre-condicionados es usado para resolver aproximadamente e iterativamente la correspondiente ecuación normal (9), aunque formándola explícitamente.

A diferencia de los algoritmos de mínimos cuadrados de mediana escala, los de larga escala pueden resolver problemas con constricciones. En este caso una dirección modificada de *Gauss-Newton* es calculada, tomando las constricciones en cuenta. También, en este caso, el Jacobiano puede ser especificado por el usuario o aproximado con una diferencia finita de cálculos (Teughels, 2003).

2.4.2.2 *lsqnonlin* en MATLAB

La función *lsqnonlin* del *toolbox* de optimización de MATLAB es la función de optimización a emplear en esta investigación debido a la experiencia obtenida en anteriores investigaciones, realizadas sobre la base de este mismo tipo de problema: *Model Updating* de modelos mecánicos de puentes

La función *lsqnonlin* como fue definida anteriormente, resuelve por mínimos cuadrados problemas no lineales de funciones o de ajuste de datos, lo que se corresponde con el tipo de problema al que esta investigación se enfrenta. A través de la misma se realiza la optimización mediante el ajuste de datos o valores obtenidos del software SAP2000 empleando la interface OAPI realizada con MATLAB.

La ecuación queda definida de la siguiente forma:

$$min_x \|f(x)\|_2^2 = min_x \sum_i f_i^2(x) \qquad (12)$$

Tal que lb≤x≤ub.

O replanteada en término de vectores.

$$min_x \|f(x)\|_2^2 = min_x (f_1(x)^2 + f_2(x)^2 + \ldots + f_n(x)^2) \qquad (12.1)$$

Con límites opcionales inferior (lb) y superior (ub) sobre los componentes de x.

Sintaxis:

- x = lsqnonlin (fun, x0)
- x = lsqnonlin (fun, x0, lb,ub)
- x = lsqnonlin (fun, x0, lb,ub,options)
- x = lsqnonlin(problem)
- [x,resnorm] = lsqnonlin(...)
- [x,resnorm,residual] = lsqnonlin(...)
- [x,resnorm,residual,exitflag] = lsqnonlin(...)
- [x,resnorm,residual,exitflag,output] = lsqnonlin(...)
- [x,resnorm,residual,exitflag,output,lambda] = lsqnonlin(...)
- [x,resnorm,residual,exitflag,output,lambda,jacobian] = lsqnonlin(...)

Descripción:

En lugar de calcular el valor $\|f(x)\|_2^2$ (La suma de los cuadrados), lsqnonlin requiere la función definida por el usuario para calcular el vector de función con valores de:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} \qquad (12.2)$$

Es decir, en términos de vectores, se puede repetir este problema de donde x es un vector o una matriz y f(x) es una función que devuelve un valor del vector.

Explicación de la sintaxis:

- x = *lsqnonlin* (fun, x0) comienza en el punto x0 y encuentra un mínimo de la suma de los cuadrados de las funciones descritas en *fun*. Se devuelve en *fun* un vector de valores y no la suma de los cuadrados de los valores. (El algoritmo implícitamente suma y eleva al cuadrado fun(x)).
- x = *lsqnonlin* (fun, x0, lb, ub) define un conjunto de límites inferior y superior en las variables de diseño en x, por lo que la solución siempre está en el rango de lb ≤ x ≤ ub. Cuando no existan límites se pasan matrices vacías en lb y ub.

- x = *lsqnonlin* (fun, x0, lb, ub, options) reduce al mínimo con las opciónes de optimización especificadas en la estructura de options. Se utiliza *optimset* para establecer estas opciónes. Para este trabajo se va utilizar la optimización mediante el Jacobiano definido por el usuario en *fun*. La función *fun* debe devolver, en el segundo argumento de salida, el valor del Jacobiano en una matriz J.

J = Jacobiano de la función evaluada en el último valor de x.

Si *fun* devuelve un vector (matriz) de m componentes y x tiene una longitud n, donde n es la longitud de x0, el Jacobiano J es una matriz de m -por- n donde J (i, j) es la derivada parcial de F(i) con respecto a x(j). El Jacobiano J es la transpuesta del gradiente de F, es decir, la metodología que se puede emplear para determinar los valores óptimos de la función que se requiere minimizar, es aproximar los valores de su gradiente a cero.

- x = *lsqnonlin* (*problem*) encuentra el mínimo del problema, donde el problema es una estructura descrita en argumentos de entrada. Crea la estructura del problema exportando un problema de Optimization APP.
- [x, *resnorm*] = *lsqnonlin* (...) devuelve el valor de la potencia 2 del residuo en x: suma (fun (x).^2).
- [x, *resnorm*, *residual*] = *lsqnonlin* (...) devuelve el valor residual de *fun*(x) en la solución x.
- [x, *resnorm, residual, exitflag*] = *lsqnonlin* (...) devuelve un valor *exitflag* que describe la condición de salida. Los posibles valores de *exitflag* y las condiciones de salida correspondientes son:

1 *lsqnonlin* converge a una solución.
2 El cambio en x fue más pequeño que la tolerancia especificada.
3 El cambio en los residuos fue más pequeño que la tolerancia especificada.
4 Dirección de búsqueda de cómputo muy pequeña.
0 Demasiadas evaluaciones de funciones o iteraciones.
-1 Parado por *output/plot* de la función.
-2 Los límites son incongruentes.
-4 La búsqueda de línea no podía reducir suficientemente el residual a lo largo de la dirección de búsqueda actual.

- [x*, resnorm, residual, exitflag, output*] = *lsqnonlin* (...) devuelve una estructura output que contiene información acerca de la optimización.
- [x*, resnorm, residual, exitflag, output, lambda*] = *lsqnonlin* (...) devuelve una estructura lambda cuyos campos contienen los multiplicadores de Lagrange en la solución x.

- [x, *resnorm, residual, exitflag, output, lambda, jacobian*] = *lsqnonlin* (...) devuelve el Jacobiano de *fun* en la solución x.

Si los límites de entrada especificados por un problema son incompatibles, la salida de x es x0 y las salidas *resnorm* y residuales son []. Los componentes de x0 que violan los límites lb ≤ x ≤ ub se restablecen en el interior del rango definido por los límites. Los componentes con respeto a los límites no se modifican. Todo lo anterior explicado se encuentra en (Afonso, 2016) y en las ayudas que posee el MATLAB (*Help*).

2.4.3 Uso del StaBIL-2.0

En este caso el uso del StaBIL-2.0: *toolbox* para el análisis por MEF de estructuras, está relacionado con la programación de los ficheros a partir del llamado que hacen un grupo de funciones programadas en estos a otras que se encuentran en la biblioteca del StaBIL-2.0. Es en este caso es donde es necesario incorporar a MATLAB esta biblioteca en caso de que no se encuentre anteriormente en el *Path* del programa.

Estas funciones son las encargadas de definir el grado de libertad con que se trabaja, reconoce el uso eficiente de matrices y sus definiciones, admite el trabajo con los valores reales de los modos y de las frecuencias, etc. Todo esto ocurre con la correcta programación de dichas funciones en los *scripts* y que se incorpore la biblioteca del StaBIL-2.0 a MATLAB.

Es el caso de ***eigfem_mdof.m*** el cual es utilizado para obtener los parámetros modales, también ***asmkm_mdof.m*** desde el cual se obtienen las matrices de masa (M) y rigidez (K) las que serán modificadas para su uso en el Método de Nelson para el Jacobiano en *on*. Se hace uso de ***mac***, ***selectdof***, ***removedof***, etc. las cuales en caso de que no se encuentren en el *Path* del programa se dificulta la corrida en MATLAB de nuestro algoritmo.

2.4.4 Declaración de los parámetros que intervienen en la optimización

En todo problema deben declararse tanto las variables de diseño, las variables de estado, la función objetivo como las restricciones del problema. Las variables de diseño (independientes) a tener en cuenta en esta investigación son las propiedades de los materiales; en dependencia del material empleado (hormigón y/o acero):

- Propiedades de peso por unidad de volumen: Densidad.
- Propiedades Isotrópicas: El módulo de elasticidad (E), el módulo de rigidez al cortante (G), el coeficiente de Poisson (μ) y el coeficiente de dilatación térmica.

- Propiedades mecánicas: La resistencia específica del hormigón (f´c) en el caso de elementos estructurales de hormigón armado o el límite de fluencia (Fy) y la resistencia última (Fu) para el caso de elementos metálicos.
- Condiciones de apoyos: para el caso de que se asígnen apoyos elásticos, la rigidez k de los mismos podría determinarse al finalizar la optimización.
- Se pueden incluir variables como los espesores de los elementos componentes del modelo estructural, determinándose así la reducción de los mismos.

Las variables de estado (dependientes) a tener en cuenta en esta investigación son los parámetros modales como:

- Las frecuencias naturales de oscilación de las formas modales seleccionadas, las coordenadas modales de estas últimas en los puntos de interés, así como la curvatura modal en cada uno de ellos.

Las Restricciones a tener en cuenta proporcionan los límites que condicionan el problema, debido a que las variables independientes no pueden tomar cualquier valor ya que están conformadas con las propiedades de los materiales de los elementos estructurales.

Se definen límites inferiores, superiores y en algunos casos pudieran emplearse restricciones de desigualdad. Estas restricciones facilitan que las respuestas obtenidas durante la calibración sean lógicas y evitará la realización de corridas adicionales.

Es también necesario conocer un grupo de elementos y conceptos que intervienen en la programación, como son:

- Coeficientes de confianza «*weighing factors*»: Los factores de peso α y β reflejan el peso señalado (normalizado) para los valores propios y los residuos de las formas modales.
- *Cross-validation*: El resultado de la actualización del modelo generalmente depende de las formas de frecuencias naturales y de modo para las que se cuenta. Excluye algunas observaciones (frecuencias naturales y/o las formas modales) y compara éstas con las características modales sacadas de la actualización del modelo. Permite la validación del modelo y del proceso de calibración.
- Posición de los sensores: Se utiliza según sea el estudio deseado (0.01, 0.02, 0.03, 0.04, 0.05, 0.06) para la dirección en que serán medidos, se declaran las condiciones de fronteras determinadas por el uso de apoyos y las matrices a utilizar para la selección de estos.
- *Eigenfrequencies* (frecuencias propias): Son indispensable, representan cantidades absolutas, provee información global, pueden estar medidas

exactamente, pueden ser muy sensible para los cambios en la rigidez, pero el número de frecuencias propias medidas es limitado.

- *Mode shapes* (formas modales): Provee información espacial, necesario para una estructura simétrica. Hace el mínimo global más pronunciado, son difíciles para medir exactamente, tienen menor sensibilidad para la rigidez del elemento.

2.4.5 Procedimiento para la obtención de los parámetros de respuesta dinámica del modelo del puente

Los parámetros modales del modelo mecánico del puente se obtienen en MATLAB a través de la OAPI SAP2000 para su posterior utilización en el procedimiento de optimización. Primero se deben declarar las variables que estarán presentes en los comandos a ejecutar en SAP2000 definidas anteriormente.

La OAPI SAP2000 dispone de numerosas funciones para trabajar con análisis modal, dentro de estas las que más interesan están vinculadas a la extracción de parámetros modales de la estructura objeto de estudio.

Para el caso del uso de la matriz Jacobiana en *on* existen dos variantes generales para su uso. La primera variante es el procedimiento general a seguir en esta investigación para su desarrollo en este capítulo y su aplicación en el próximo.

Se definen, a continuación, los pasos generales que se tienen en cuenta para el uso del Jacobiano en *on* pero que no será implementado completamente en esta investigación por la complejidad que presenta a la hora de establecer un algoritmo seguro y exacto:

Variante 1: SAP2000-OAPI-MATLAB:

1. Crear el modelo del puente en SAP2000 que posee extensión *.sdb y una carpeta que contenga dicho modelo (Esta carpeta es la dirección para ejecutar el modelo desde MATLAB).
2. Introducir en esta carpeta la biblioteca StaBIL-2.0 (la misma contiene funciones necesarias para el uso de los *scripts* que se confeccionarán) en caso de que no se encuentre en el *Path* del programa.
3. Ejecutar MATLAB R2015a.
4. Confeccionar los *scripts* (.m) La base de estos ficheros está en:
   I. Generar matriz K (rigidez) y M (masa). Extraer K y M.
   II. Medir sensibilidad de dK y dM.
   III. Determinar dω y dø a través del Método de Nelson.
   IV. Calcular el Jacobiano.
   V. Implementar en la calibración.

Variante 2: Combinación entre SAP2000 y StaBIL-2.0:

1. Crear el modelo del puente en SAP2000 que posee extensión *.sdb y una carpeta que contenga dicho modelo (Esta carpeta es la dirección para ejecutar el modelo desde MATLAB). En este caso se hace uso del fichero de extensión *.$2k para la obtención de la información necesaria que contiene el modelo con que se trabaja para su uso en StaBIL-2.0.
2. Introducir en esta carpeta la biblioteca StaBIL-2.0 (la misma contiene funciones necesarias para el uso de los *scripts* que se confeccionarán) en caso de que no se encuentre en el *Path* del programa.
3. Generar geometría y conectividad con StaBIL-2.0 con funciones en MATLAB.
4. Completar el modelo en StaBIL-2.0 con la ayuda del fichero *.$2k.
5. Implementar calibración similar a la explicación que le sigue para matriz Jacobiana en *off/on* (procedimiento desarrollado en esta investigación).

A continuación, se define una serie de pasos específicos del procedimiento implementado en MATLAB para el uso de la matriz Jacobiana en *off-on* para esta investigación:

1. Crear una carpeta con el nombre deseado donde se incluya lo siguiente:
   a) La creación del modelo del puente en SAP2000 que posee extensión *.sdb y una carpeta que contenga dicho modelo (Esta carpeta es la dirección para ejecutar el modelo desde MATLAB). Este modelo debe ser utilizado como referencia, en el mismo es donde se trabaja con los daños inducidos o los valores reales de las mediciones realizadas a la estructura. Ejemplo de nombre: Ref_Model.
   b) La elaboración del modelo del puente en SAP2000 con extensión *.sdb en este caso sin daño inducido con el cual se calcularán los parámetros actualizados. Ejemplo de nombre: Upd_Model.
   c) Introducir en esta carpeta la biblioteca StaBIL-2.0 (la misma contiene funciones necesarias para el uso de los *scripts* que se confeccionarán) en caso de que no se encuentre en el *Path* del programa.
2. Para el trabajo con MATLAB R2015a.

Matriz Jacobiana en *off*:

   a) Confeccionar los *scripts* (.m) con los nombres siguientes (***data_bridge, eigfem_bridge, main_bridge, sensitivity_analysis, upd_bridge***) donde se escribirán los comandos OAPI a ejecutar en MATLAB para interactuar con el modelo de SAP2000v.16 que se encuentra en la carpeta creada en el paso 1. (Estos comandos son formulados según la documentación OAPI que permite la interface).

Matriz Jacobiana en *on*:

b) Confeccionar los *scripts* (.m) con los nombres siguientes (***data_bridge, run_SAP2k, main_bridge, sensitivity_analysis, upd_bridge, asmkm_sap2k.m***) donde se escribirán los comandos OAPI a ejecutar en MATLAB para interactuar con el modelo de SAP2000v.16 que se encuentra en la carpeta creada en el primer paso.

En ambos casos se debe programar las matrices necesarias para almacenar los resultados de los modos identificados para obtener dichos resultados directamente en la línea de comandos, se recomienda en este caso crear una matriz para almacenar las frecuencias naturales de los modos identificados y otra para almacenar las coordenadas modales de los puntos antes analizados.

Es importante obtener estos parámetros en la línea de comandos porque son los que hacen posible, una vez cumplidos estos pasos, realizar el procedimiento para calibrar el modelo mediante la optimización y al finalizar la corrida MATLAB guarda toda la información en su *Workspace*.

2.4.6 Procedimiento para la calibración de modelos estructurales. Jacobiano en *off*

El procedimiento seguido en MATLAB para la optimización consta de la creación de los siguientes *scripts* o ficheros: (***data_bridge, eigfem_bridge, main_bridge, sensitivity_analysis, upd_bridge***) los cuales trabajan en interacción para llevar a cabo todo el proceso de actualización. Estos están confeccionados con lenguaje de programación, presentándose solo acá un resumen de la confección de los mismos con la base principal de los códigos utilizados, siendo estos necesarios para el análisis correcto del procedimiento de calibración. Se remite a los anexos correspondientes donde aparece la programación completa.

2.4.6.1 Definición del modelo de referencia (*data_bridge.m*)

En este *script* se define el modelo de referencia con el cual se va a trabajar desde MATLAB, pero elaborado en SAP2000v16 con extensión *.sdb. Se establecen los valores de los parámetros de referencia para calibrar el modelo mecánico de la estructura: coordenadas modales de las frecuencias por oscilaciones propias, frecuencias circulares correspondientes y los grados de libertad, en general proporciona las propiedades reales las cuales el modelo en Sap2000 debe presentar al final de la calibración.

Los pasos para trabajar son (Anexo 1):

1. *Close all, clear all, clc*: Para limpiar y cerrar todo para empezar con el nuevo trabajo, se limpia el *workspace*, el *command Windows* y cualquier figura o gráfico que se esté abierto.
2. Determinar las secciones o elementos/miembros a analizar (se define la cantidad) para el caso de la actualización de los parámetros (*n_floors, n_elements, n_sections, n_upd_param*).
3. Determinar el valor real de la variable a obtener (*x_ref*) como valor de referencia.
4. Declarar la dirección exacta del modelo (*file*) que se utiliza en SAP2000v16 con extensión *.sdb para su uso en ***eigfem_bridge.m*** donde se utilizan los comandos OAPI. En este caso sería el modelo nombrado como Ref_Model.
5. Declarar el uso de los ficheros donde se obtiene y se trabaja con los datos de referencia para extraer los parámetros modales de referencia:
   - *[phi_ref,omega_ref, DOF_ref] = eigfem_bridge(x_ref,file);*
6. Declarar el número de nodos, modos y DOF, se calculan los parámetros modales para estos modos.
7. Salvar los datos modales para la actualización y los parámetros del sistema de referencia en archivos con extensión *.mat: *data_bridge phi_ref omega_ref DOF_ref n_upd_param.*

2.4.6.2 Corrida de SAP2000 y obtención de los parámetros modales *(eigfem_bridge.m)*

La utilización de este *script* se basa en la corrida del modelo de la estructura objeto de estudio de extensión *.sdb creado en SAP2000v16 con propiedades asignadas (E) que se le es transferida a partir del parámetro formal *x-moduli*, ejecutando en este caso el *Load Case* '*MODAL*' lo que permite la obtención de los parámetros modales necesarios. Este fichero es utilizado por ***upd_bridge.m*** para la actualización del modelo, el trabajo con los valores reales de los modos y de las frecuencias en modelos de elementos finitos.

En este caso los pasos para la confección del *script* quedan definidos de la siguiente forma (Anexo 2):

1. Definir la función a utilizar: *[phi,omega,DOF] = eigfem_bridge(x_moduli,file)* con su estructura de control.
2. Escribir comandos OAPI:

a) *feature('COM_SafeArraySingleDim', 1)*: Favorece el paso de los datos a SAP2000v16 como matrices de dimensión 1.

b) *feature('COM_PassSafeArrayByRef', 1)*: Permite pasar las matrices no escalares por referencia.

c) *SapObject = actxserver('sap2000v16.SapObject')*: Para crear el objeto SAP2000v16.

d) *SapObject.ApplicationStart*: Para iniciar la aplicación de SAP2000v16.

e) *SapObject.Hide*: Para que el SAP2000 no salga en pantalla y trabaje en segundo plano.

f) *SapObject.SapModel.File.OpenFile*: Para abrir el modelo estructural del puente en SAP2000 según la ruta especificada (se declara la dirección exacta del modelo que se utiliza en SAP2000v16 con extensión *.sdb).

g) *SapModel = SapObject.SapModel*: Para crear el objeto SapModel, a partir del mismo se pueden escribir los comandos omitiendo SapObject.

h) *SapModel.SetModelIsLocked(0)*: Para desbloquear el modelo, este comando es opciónal en dependencia de si se quiere cambiar algo en el modelo desde MATLAB.

i) *SapModel.SetPresentUnits*: Selección de las unidades de trabajo

j) *SapModel.Analyze.CreateAnalysisModel*: Para crear el caso de análisis.

k) *SapModel.PropMaterial.SetMaterial*: Definición de los materiales creados.

l) *SapModel.PropMaterial.SetMPIsotropic*: Asignar las propiedades mecánicas isotrópicas.

m) Asignar otras propiedades como (*f´c IsLightweight fcsfactor eFu SSType SSHysType StrainAtfc StrainUltimate FrictionAngle DilatationalAngle Temp*)

n) *SapModel.FrameObj.SetMaterialOverwrite*: Sobreescribir las propiedades en los elementos seleccionados del modelo estructural de la superestructura. Esto se realiza para cada tramo, elemento o sección que se este analizando.

o) *SapModel.Analyze.SetRunCaseFlag*: Seleccionar el caso de carga.

p) *SapModel.Analyze.SetSolverOption_1*: Ajusta la opción del SAP2000 *Solver Options* y obtener en la corrida los ficheros de extensión *.TXA, *.TXE, *.TXM, *.TXK, *.TXC y obtener entonces las matrices de masa y rigidez en caso de ser necesarios o que el usuario quisiera conocer estos datos.

q) *SapObject.SapModel.Analyze.RunAnalysis()*: Para correr el análisis.

r) *SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput:* Se usa para deseleccionar todos los casos y combinaciones de carga.

s) *SapModel.Results.Setup.SetCaseSelectedForOutput('MODAL'):* Seleccionar el análisis Modal que es el que interesa.

t) *SapModel.Results.ModalPeriod(NumberResults, LoadCase, StepType, StepNum, Period, Frequency, CircFreq, EigenValue*): Este comando es el encargado de extraer los parámetros modales de frecuencias y períodos de oscilación así como los valores propios. Cuenta con:

I. Comandos para obtener las frecuencias para cada modo de oscilación.
II. Se define la variable omega como las frecuencias naturales.
III. Comandos para obtener las coordenadas modales en x, y y z.

u) *SapModel.Results.ModeShape('ALL', GroupElm, NumberResults, Obj, Elm, ACase, StepType, StepNum, U1, U2, U3, R1, R2, R3):* Permite extraer del modelo las formas modales de todos los modos de oscilación del mismo o del modo especificado. Posee la siguiente estructura:

I. Se definen las coordenadas modales en x, y y z como phix, phiy y phiz respectivamente.
II. Se calculan el número de modos.
III. Se calculan el número de nodos de la estructura.
IV. Se definen matrices intermedias para obtener las coordenadas modales definitivas.
V. Se calculan las coordenadas modales definitivas.
VI. Se crea una matriz phi que contiene las coordenadas modales en las tres direcciones principales.
VII. Se definen los grados de libertad de la estructura en las tres direcciones.

v) *SapObject.ApplicationExit:* Cerrar la aplicación SAP2000v16.

2.4.6.3 Inicialización de parámetros (*main_bridge.m*)

Este fichero posee gran importancia pues es el que se corre desde MATLAB en interacción con los demás ejecutando cada una de las funciones existente. Es el fichero principal pues realiza la actualización del modelo mecánico de la estructura basado en los valores de referencia (frecuencias y coordenadas de las formas principales de oscilaciones propias). El mismo es el encargado de iniciar el trabajo con los parámetros modales, lleva a cabo el ploteo de los resultados finales e interactúa directamente con ***upd_bridge.m*** en la actualización del modelo. Como respuesta se van presentando en el *Command Windows* de MATLAB los números de iteraciones que se han ejecutado y los residuos para cada una de ellas.

Pasos en la confección del fichero (Anexo 3):

1. Definir el modelo de referencia a ejecutar, correr dicho modelo y guardar los datos de referencia. Llama a correr o ejecutar *data_bridge.m*
2. Establecer los parámetros a actualizar iniciales (*E_0, E_min, E_max*) como los límites superiores e inferiores y los valores iniciales para la optimización.
3. Fijar pesos asígnados al error (*alpha* α*, beta* β) de las frecuencias propias de oscilación y de las formas modales.

4. Establecer como salvar: *upd_param_bridge alpha beta E_0*
5. Se carga del archivo .mat creado al ejecutar el script *data_bridge*: *data_bridge phi_ref omega_ref DOF_ref n_upd_param.*
6. Definir la función a utilizar fijando los valores de variación para las iteraciones, es decir límites superiores e inferiores y el punto de partida para la calibración (*T_0, T_min, T_max, T_upd, E_upd*).
7. Ejecutar la actualización o nueva corrida con los parámetros de la optimización *[T_upd]* = upd_mdof(T_0,T_min,T_max).
8. Se calcula al finalizar la optimización los valores de las variables de diseño calibradas (E_upd). Aquí termina el proceso de la calibración.

Presentación de los resultados:

1. Calcular los parámetros modales iniciales en la iteración a partir de la declaración de la dirección del modelo a actualizar (sin daños). En este caso sería el modelo nombrado como Upd_Model. Obtener :
   a. *[phi_0,omega_0,DOF_0] = eigfem_bridge(E_0,file)*
2. Calcular los parámetros modales actualizados en la iteración.
   a. *[phi_upd,omega_upd,DOF_upd] = eigfem_bridge(E_upd,file)*
3. Definición de la ubicación de los sensores según sea el análisis deseado proponiendo los nodos deseados y (0.01, 0.02, 0.03, 0.04, 0.05, 0.06) para la dirección en que serán medidos, condiciones de fronteras y las matrices a utilizar. Calcular la matriz L de referencia o inicial.
4. Crear un resumen de resultados para obtener el Módulo de deformación, frecuencias naturales y rigidez, teniendo en cuenta:

a) Programar las matrices de recopilación de datos de los resultados (Datos de la actualización del modelo, valores de referencias y de comparación, porcientos de diferencias) para las variables de diseño iniciales y calibradas, las frecuencias iniciales, calibradas y de referencia.
b) Uso de la matriz MAC inicial y actualizada.
c) Imprimir los resultados por pantalla.
d) Cargar los resultados de las iteraciones.

5. Definir las caracteristicas del gráfico con resumen de las iteraciones realizadas en relación con la función objetivo (norma de residuos) para el ploteo de los resultados.
6. Definir las caracteristicas del gráfico con resumen del trabajo de los modos analizados.

2.4.6.4 Actualización del modelo (*upd_bridge.m*)

El fichero ***upd_bridge.m*** es el encargado de:

- Especifica las opciónes para actualizar (***optimset.m***)
- Define la función objetivo (@***objfun***)
- Calcula los parámetros modales en interacción con (***run_SAP2k***)
- Definición de la función de salida (@***outfun***)
- Permite la optimización no lineal utilizando (***lsqnonlin.m***)
- Incorpora en sus funciones de actualización la matriz Jacobiana en (***off-on***).

Pasos en la confección del fichero (Anexo 4):

1. Definir la función objetivo: *[T_upd]=upd_mdof (T_0,T_min,T_max)* **.**
2. Definir las restricciones (tolerancias) en la función objetivo y los parámetros a actualizar. Especificar la opción a utilizar para la actualización: Se definen las opciones que tendrá la optimización, el algoritmo a emplear, algunas características propias de estos algoritmos, se define si la matriz Jacobiana será suministrada o no, si se desea ir mostrando los valores de cada iteración, se definen las tolerancias y la función de salida.

En este caso es posible el uso de diferentes métodos y se define el uso o no de la matriz Jacobiana (***'on'*** o ***'off'***):

- ***Trust-Region-Reflective Least Squares Algorithm***

Caso 1:

% Specify options
*options=optimset('Display','iter','TolFun',TolFun,'TolX',TolX,...'Jacobian',**'off'**,'OutputFcn', @outfun);*

- Definir el uso de la optimización no lineal, la función objetivo a utilizar, el valor inicial y las constricciones y las opciones de optimización.

[T_upd,resnorm,residual,exitflag,output]=...lsqnonlin(@objfun,T_0,T_min,T_max,options);

Caso 2:

% Specify options
*options=optimset('Display','iter','TolFun',TolFun,'TolX',TolX,...'Jacobian',**'on'**,'OutputFcn', @outfun);*

- Definir el uso de la optimización no lineal, la función objetivo a utilizar, el valor inicial y las constricciones y las opciones de optimización.

[T_upd,resnorm,residual,exitflag,output]=...lsqnonlin(@objfun,T_0,T_min,T_max,options);

- ***Levenberg-Marquardt Method***

Como el otro algoritmo para la calibración tiende a no encontrar el mínimo correctamente o llegar a un punto donde los residuos se mantienen constantes y la optimización se detiene debido a que la distancia del paso es muy pequeña, por debajo de la tolerancia permitida o supera el número de iteraciones posibles, se puede utilizar el siguiente. Se definen límites a las variables de diseño de -infinito y +infinito, debido a que este algoritmo de optimización no permite que se le definan restricciones al problema.

% Specify options

*options=optimset('Algorithm','levenberg-marquardt','ScaleProblem','Jacobian','Display','iter-detailed',...'Jacobian','**off/on**','TolFun',TolFun,'TolX',TolX,'OutputFcn',@outfun);*

% Nonlinear Optimization

[T_upd,resnorm,residual,exitflag,output]=...

*lsqnonlin(@objfun,T_0,-inf*ones(1,n_upd_param),inf*ones(1, n_upd_param),options);*

Para continuar la explicación del diseño del fichero se continúa con los siguientes pasos que serían los mismos para ambos casos.

Caso 1: Matriz Jacobiana en ***'off'***

a) Definir la función objetivo donde se obtienen los residuos por frecuencias y formas modales: *[res,jac]=objfun(T)*. En este caso se define además:

I. Cargar: *upd_param_bridge alpha beta E_0*. Definir *Evec* como variable de diseño para cada iteración.

II. Cargar datos modales experimentales: *data_bridge phi_ref omega_ref DOF_ref n_upd_param.*

III. Calcular los parámetros modales utilizando el modelo del puente, se define la dirección de dicho modelo (*file*). En este caso sería el modelo sin daños nombrado como Upd_Model.

IV. Se define el caso a utilizar en la actualización. Ya que el código no definía el uso de la OAPI se incorpora un caso 3 (anteriormente contaba con *Analytical Model* y *Ansys Model*), dirigiendo la actualización al uso de la OAPI SAP2000-MATLAB. En este caso se calculan los parámetros dinámicos para cada iteración.

Case 3: *OAPI SAP2000 MatLab MODEL*

- *[phi_i,omega_i,DOF_i] = eigfem_bridge(Evec,file);*

V. Definición de la ubicación de los sensores según sea el análisis deseado proponiendo los nodos deseados y (0.01, 0.02, 0.03, 0.04, 0.05, 0.06) para la

dirección en que serán medidos, condiciones de fronteras y las matrices a utilizar. Calcular la matriz L de referencia o inicial.

VI. Trabajo con los modos:

a) Calcular la matriz MAC. Definir rango de análisis para los modos (*treshold*). Se calcula la matriz MAC para las frecuencias modales de referencia y las calculados en cada iteración. En el caso que el valor más grande de la matriz MAC obtenida sea menor que el valor *treshold* o límite se presenta en el C*ommand Windows* que no se ha encontrado ninguna coincidencia para los modos medidos.
b) Establecer la estructura de control para el trabajo con los modos.
c) Se calculan las formas y las frecuencias modales de referencia y las calculadas en cada iteración para los modos seleccionados. También se definen el número de modos.
d) Se calculan los factores de escala para las formas modales.
e) Calcular los valores propios (*eigenvalues*).
f) Calcular los residuos de la función objetivo para las frecuencias y las formas modales en los modos seleccionados.

VII. Definir la función: *stop = outfun(T,optimValues,state)*

a) Salvar los resultados de las iteraciones en un archivo .mat para cada iteración, los que se van acumulando hasta que termina la optimización.
b) Establecer la estructura de control de la función.

2.4.6.5 Análisis de sensibilidad (*sensitivity_analysis.m*)

En el análisis de sensibilidad se tiene en cuenta el uso de la matriz MAC, la misma posibilita la validación del modelo y la calibración de este. Este fichero es el encargado de hacer el análisis de esta matriz, el cual tiene como finalidad lograr mediante la confección de un gráfico que en la diagonal principal de la misma se logre los valores más cercanos posible a 1. Esta cercanía a 1 implica una correlación perfecta entre las formas modales del modelo numérico y las formas modales de la estructura real.

Este fichero es el mismo para el caso del análisis que se vaya a llevar a cabo tanto con el uso de la matriz Jacobiana (*'on'* o *'off'*) y los métodos de optimización *Trust-Region-Reflective Least Squares Algorithm* y *Levenberg-Marquardt Method*.

La confección de este *script* tiene los siguientes pasos (Anexo 5):

1. Establecer las restricciones (modelo de referencia entre el daño y la estructura deseada (*E_ref / E_0).*
2. Calcular los parámetros modales definiendo la función a utilizar.

3. Seleccionar los modos para calcular los parámetros modales.
4. Determinar la posición de los sensores: (*DOF_sensors)*: La ubicación de los sensores está definida por el análisis que se desee realizar teniendo en cuenta los distintos movimientos posibles (0.01, 0.02, 0.03, 0.04, 0.05, 0.06) para los nodos deseados, condiciones de fronteras y las matrices a utilizar.
5. Seleccinar y programar las matrices en las que se reflejan los resultados de acuerdo a estos sensores *(L_ref / L_0).*
6. Calcular la matriz AUTO-MAC teniendo en cuenta los modos seleccionados.
7. Definir las propiedades del gráfico en el que se reflejan los resultados para el análisis de sensibilidad (coordenadas, colores, tipo de grafico, escalas) para llevar a cabo el análisis correspondiente.
8. Concluida la corrida hacer análisis de resultados.

2.4.7 Procedimiento para la calibración de modelos estructurales. Jacobiano en *on*

El procedimiento seguido en MATLAB para la optimización para el caso del uso de la matriz Jacobiana en *on* consta de la creación de los siguientes *scripts* o ficheros: (***data_bridge, run_SAP2k, main_bridge, sensitivity_analysis, upd_bridge, asmkm_sap2k.m***) los cuales trabajan en interacción para llevar a cabo todo el proceso de actualización y tienen similar estructura que la explicada anteriormente. Se remite a los anexos correspondientes donde aparece la programación completa. En el caso de ***sensitivity_analysis*** es el mismo para ambos casos.

2.4.7.1 Definición del modelo de referencia (*data_bridge.m*)

En este *script* se define el modelo de referencia con el cual se va a trabajar desde MATLAB, pero elaborado en SAP2000v16 con extensión *.sdb y los ficheros necesarios para el trabajo con las matrices de masa (M) y rigidez (K). Igualmente se establecen los valores de los parámetros de referencia para calibrar el modelo mecánico de la estructura: coordenadas modales de las frecuencias por oscilaciones propias, frecuencias circulares correspondientes y los grados de libertad, se calcula además los parámetros modales $\omega i, \emptyset$ con la interacción con el fichero ***eigfem*** de StaBIL-2.0. Los pasos para trabajar son (Anexo 6):

1. 2. Los dos primeros pasos a seguir son igual que el de ***data_bridge*** para el Jacobiano en *off*.
3. Declarar la dirección exacta del modelo (*file-filex*) que se utiliza en SAP2000v16 con extensión *.sdb para el caso de su uso en ***run_SAP2k.m*** donde se utilizan los

comandos OAPI: *run_SAP2k(x_ref,file)* y *.TXA para el uso de *asmkm_sap2k(filex)* en la obtención de las matrices de masa y rigidez. En este caso sería el modelo nombrado como Ref_Model.

4. Declarar el número de nodos, modos y DOF.
5. Declarar el uso de los ficheros donde se obtiene y se trabaja con las matrices de masa y rigidez principalmente y los datos de referencia:

[K_ref,M_ref,Tt_0]=asmkm_sap2k(); [phi_ref,omega_ref] = eigfem(K_ref,M_ref,n_m);

6. Salvar los datos modales para la actualización y los parámetros del sistema de referencia en archivos con extensión *.mat: *data_bridge phi_ref omega_ref DOF_ref n_upd_param.*

2.4.7.2 Corrida de SAP2000 *(run_SAP2k.m)*

La utilización de este *script* se basa en la corrida del modelo de la estructura objeto de estudio de extensión *.sdb creado en SAP2000v16 con propiedades asignadas (E) que se le es transferida a partir del parámetro formal *x-moduli*, ejecutando el *Load Case* '*DEAD-MODAL*' lo que permite la obtención de los ficheros de extensión *.TXA, *.TXE, *.TXM, *.TXK, *.TXC necesarios para obtener la matriz K y M. Este fichero es utilizado por ***upd_bridge.m*** para la actualización del modelo, el trabajo con los valores reales de los modos y de las frecuencias en modelos de elementos finitos. En este caso los pasos para la confección del *script* quedan definidos de la siguiente forma (Anexo 7):

1. Definir la función a utilizar: : *[] = run_SAP2k (x_moduli,file)* con su estructura de control.
2. Escribir comandos OAPI: En este caso se utilizan los mismos que para ***eigfem_bridge*** en el Jacobiano en *off* incorporándose:

a) *SapModel.Analyze.SetSolverOption_1*: Ajusta la opción del SAP2000 *Solver Options* y obtener en la corrida los ficheros de extensión *.TXA, *.TXE, *.TXM, *.TXK, *.TXC y obtener entonces las matrices de masa (M) y rigidez (K).

2.4.7.3 Inicialización de parámetros (***main_bridge.m***)

Este fichero posee la misma importancia que su similar pues es el que se corre desde MATLAB en interacción con los demás ejecutando cada una de las funciones existente. Es el fichero principal pues realiza la actualización del modelo mecánico de la estructura basado en los valores de referencia (frecuencias y coordenadas de las formas principales de oscilaciones propias). Posee la misma estructura que el ***main_bridge*** para el Jacobiano en ***off***. Para la confección del fichero solo cambia cómo calcular los parámetros modales nominales y actualizados (Anexo 8):

1. Calcular los parámetros modales iniciales en la iteración a partir de la declaración de la dirección del modelo a actualizar (sin daños). En este caso sería el modelo nombrado como Upd_Model. Obtener:
   b. *run_SAP2k(E_0,file)*
   c. *[K_0,M_0,T_0,DOF_0]=asmkm_sap2k(filex);*
   d. *[phi_0,omega_0] = eigfem(K_0,M_0)*
9. Calcular los parámetros modales actualizados en la iteración.
   b. *run_SAP2k(E_upd,file)*
   c. *[K_upd,M_upd,T_upd,DOF_upd]=asmkm_sap2k(filex);*
   d. *[phi_upd,omega_upd] = eigfem(K_upd,M_upd)*

2.4.7.4 Actualización del modelo (*upd_bridge.m*)

El fichero ***upd_bridge.m*** es similar al del uso del Jacobiano en *off* solo incorporándose en este caso el uso del Método de Nelson y el trabajo con las matrices de masa (M) y rigidez (K). A continuación, se mencionan los cambios con respecto a ***upd_bridge.m*** utilizado en Jacobiano en *off*: (Anexo 9)

1. Calcular los parámetros modales utilizando el modelo del puente, se define la dirección de dicho modelo (*file-filex*) tanto para la extensión *.sdb como para la obtención de las matrices de masa (M) y rigidez (K). En este caso sería el modelo nombrado como Upd_Model.
2. Se define el caso a utilizar en la actualización. Ya que el código no definía el uso de la OAPI se incorpora también un caso 3 dirigiendo la actualización al uso de la OAPI SAP2000-MATLAB en este caso.

   Case 3: *OAPI SAP2000 MATLAB MODEL*

   - *run_SAP2k(Evec,file);*
   - *[K_i,M_i,~,DOF_i]=asmkm_sap2k(filex);*
   - *[phi_i,omega_i] = eigfem(K_i,M_i);*

3. Calcular la matriz Jacobiana (*Nelson's Method*):

   a) Definir el número de parámetros a actualizar (*npar*).

   b) Definir el número de sensores (*nd*).

   c) Definir el grado de libertad del modelo (*ndof*).

   d) Determinar las derivadas dK y dM para la cantidad de secciones a actualizar de la siguiente forma:

   – *Calculate the derivative of the stiffness matrix w.r.t the Updating parameters (Sensitivity to Updating parameters*)

   *T0 = T (ind1); delta_T = 1e-8 * T0; % Step applied for finite difference calculation*
   *xl = E_0*ones(npar,1); xl(ind1) = xl(ind1)*(T0-delta_T/2);*

*xu = E_0*ones(npar,1); xu(ind1) = xu(ind1)*(T0+delta_T/2);*

- *Run model for xl (this will create the analysis model)*
  run_SAP2k (xl, file);
- *Read stiffness and mass matrix of initial model for xl*
  [Kl, Ml, Tl, DOFl]=asmkm_sap2k (filex); nDOF=length (DOFl);
- *Run model for xu (this will create the analysis model)*
  run_SAP2k (xu, file);
- *Read stiffness and mass matrix of initial model for xu*
  [Ku, Mu, Tu, DOFu]=asmkm_sap2k (filex);
- *Derivatives dK/dx and dM/dx*
  dK = (Kl-Ku) / delta_T;
  dM = (Ml-Mu) / delta_T;
  clear Kl Ku xl xu;

e) Definir los valores propios de la matriz Jacobiana y los vectores propios de dicha matriz (*eigenvectors: jac_lambda; jac_vec*).

f) Definir el sistema de origen de las matrices (A y b) según *Nelson's method*.

g) Modificar el sitema de matrices (A* y b*) según *Nelson's method*.

h) Obtener: *jac=[jac_lambda;jac_vec].*

2.4.7.5 Obtención de matrices K, M, T (*asmkm_sap2k.m*)

Este fichero es nuevo con respecto a el procedimiento anterior el cual posee gran importancia ya que desde este *script* se exportan las matrices de masa (M), rigidez (K), *constraint* (T), el vector DOF a partir de SAP2000 y la salida de ficheros con extensión *.TXA, *.TXE, *.TXM, *.TXK, *.TXC. El procedimiento para la confección de este *script* es: (Anexo 10)

1. Cargar el archivo (*file*) con la dirección del mismo de la siguiente forma: *[pathstr,name,ext] = fileparts(file)*
2. Leer los archivos de extensión *.TXA, *.TXE, *.TXM, *.TXK, *.TXC:
   I. Cargar *.TXK: En este segmento se lee el fichero TXK que contiene la matriz rigidez de SAP2000. Se asígna a la variable K0 (matriz rigidez inicial), los datos leidos y se guarda como la matriz rigidez inicial. Se cierra el fichero abierto.
   II. Cargar *.TXM: En este segmento se lee el fichero TXM que contiene la matriz de masas de SAP2000. Se asígna a la variable M0 (matriz masa inicial), los datos leidos y se guarda como la matriz rigidez inicial. Se cierra el fichero abierto.
   III. Cargar *.TXE: En este segmento se lee el file TXE que contiene los núm. de ecuac. de SAP2000. Se asígna a la variable DOF0 (inicial). Se cierra el fichero abierto.

IV. Cargar *.TXC: En este segmento se lee el fichero TXC que contiene las *constrains* de SAP2000 y las incorpora a K. Se usa *try* para detector si hay error, puesto que puede que no haya *constrains* y en ese caso no habrá fichero *.TXC.

3. Definir la variable DOF. Se usa *reshape* ya que cambia las dimensiones de la matriz DOF0 traspuesta a una de 6 grados de libertad por nodo, como corresponde a SAP2000 esto es para análisis 3D, si fuera análisis plano seria por 3. Definir los DOF de StaBIL-2.0 y se establecen los *masters* y *slaves* de los *constrains*.
4. Genarar la matriz rigidez (K) incluyendo los Master DOF
5. Generar la matriz de masa (M).
6. Generar la matriz (T) que contiene los *constrains* en caso de que existan.
7. Salvar las matrices K, M, DOF, *masters*, *slaves* y T.
8. Borrar: *DOF0 I J K0 M0 MasterDOF Masterdof SlaveDOF Slavedof dof ext file flag name pathstr*.

En este fichero se hace uso de la función *match.m*: Función que casa los nodos de los elementos y los números de nodos que se den de dato con los correspondientes índices de los elementos en la matriz rigidez según la conectividad utilizada en el modelo. Utiliza la función de MATLAB *ismember* que devuelve 1 cuando los datos de los nodos de los elementos se encuentran en la misma posición de los números de los nodos correspondientes.

asmkm_sap2k.m ensambla la matriz M, K y T (en caso de tener *constraint* en el modelo de SAP2000):

- Carga y lee los *.TXK, *.TXM, *.TXE, *.TXC.
- Establece los DOF.
- Forma las matrices K, M, T por ese orden.
- Salva las matrices K, M, T.

2.5 Conclusiones parciales del capítulo

Confeccionado el procedimiento para la calibración de modelos estructurales utilizando la interface OAPI SAP2000-MATLAB que incluye la matriz Jacobiana se llega a las siguientes conclusiones:

- La modelación de puentes es un proceso que conlleva trabajos de campo y de oficina, lo cual permite obtener un gran número de datos para comenzar a modelar la estructura en los softwares correspondientes, demostrando que es de vital importancia poseer la mayor cantidad de información posible.
- Las variables de diseño están formadas por las propiedades isotrópicas de los materiales y el módulo de deformación, pues este es un parámetro de la rigidez de la estructura y las variables de estado son las frecuencias naturales de oscilación, las formas modales seleccionadas y las curvaturas modales de estas.
- La función objetivo (que no es más que la suma de los cuadrados del vector residual compuesto por los residuos por frecuencias naturales, los residuos por formas modales y los residuos por curvaturas modales) se minimiza empleando un algoritmo de optimización basado en el método de la región de confianza *Gauss-Newton* implementando en MATLAB la función: ***lsqnonlin*** que incluye o no la matriz Jacobiana.
- El procedimiento confeccionado en MATLAB para la calibración consta de la creación de los siguientes *scripts* para el caso del uso de la matriz Jacobiana en ***off***: (***data_bridge, eigfem_bridge, main_bridge, sensitivity_analysis, upd_bridge***) y (***data_bridge, run_SAP2k, main_bridge, sensitivity_analysis, upd_bridge, asmkm_sap2k.m***) para el uso en ***on***, los cuales trabajan en interacción para cada caso y llevan a cabo todo el proceso de actualización.
- La incorporación de ***asmkm_sap2k.m*** juega un papel importante ya que se exportan las matrices de masa (M), rigidez (K), constraint (T), el vector DOF a partir de SAP2000 y la salida de ficheros con extensión *.TXA, *.TXE, *.TXM, *.TXK, *.TXC permitiendo la aplicación del Método de Nelson para el cálculo de la matriz Jacobiana en la actualización.
- El uso del análisis de sensibilidad con la matriz MAC admite la validación del modelo y su calibración logrando una correlación perfecta entre las formas modales del modelo numérico y las formas modales de la estructura real tanto para el uso o no de la matriz Jacobiana en el proceso.

Capítulo III: Influencia de la aplicación a un caso de estudio del procedimiento automatizado para la calibración estructural utilizando o no el Jacobiano.

En este capítulo se aplica el algoritmo de calibración hecho previamente para obtener el modelo mecánico calibrado del puente objeto de estudio llevando a cabo el procedimiento para la calibración de modelos estructurales utilizando la interface OAPI SAP2000-MATLAB. Este se analiza teniendo en cuenta el tiempo de máquina empleado, la veracidad de sus resultados en cuanto a los parámetros de la respuesta dinámica, lograr la validación y calibración del modelo y un buen análisis de sensibilidad.

3.1 Introducción

El procedimiento desarrollado conlleva el uso del SAP2000v16 para la conformación del modelo y del MATLAB R2015a que interactúa mediante la OAPI SAP2000 con el software anterior para obtener los resultados del análisis modal. El modelo puede ser desarrollado desde MATLAB lo que traería consigo un largo trabajo y de escritura de códigos, por lo que se hizo en SAP2000 para interactuar mediante la OAPI y así ahorrar tiempo. El algoritmo elaborado para la calibración de puentes es aplicable a cualquier caso de estudio, teniendo en cuenta que el modelo mecánico del mismo responda en correspondencia a su comportamiento estructural en la realidad.

En esta investigación se utiliza como caso de estudio y de análisis el modelo mecánico del Puente km.9.578 de Línea Cenizas (Foto 3.1) perteneciente al tramo Estación Santa Clara–Cenizas. El Puente km.9.578 de Línea Cenizas perteneciente a la red de vías de ferrocarriles de Cuba se encuentra ubicado hacia el este a 9,5 kilómetros de la Estación de Santa Clara, sobre este cruzan todos los trenes de cargas y pasajeros que se trasladan desde o hacia las provincias orientales al centro u occidente del país, por lo que se encuentra en estos momentos en explotación. Salva una longitud de 19.2m sobre el río La Movida el cual en periodos lluviosos el agua asciende notablemente (Puentes, 2005). El mismo presenta corrosión en los nudos de apoyos, los arriostramientos y en otras partes de sus elementos, además de tener un buen grupo de traviesas en mal estado. Este puente ha sido objeto de estudio por varios autores, los cuales han trabajado desde la identificación de sus patologías, obtención de mediciones, etc., hasta su modelación y calibración (Aragón, 2010, González, 2015, Afonso, 2016, González, 2007).

Foto 3.1: *Vista General del Puente km.9.578 de Línea Cenizas en SAP2000v16.*
Fuente:(Puentes, 2005).

El modelo fue elaborado teniendo en cuenta todas sus características reales en cuanto luces, uniones, materiales y una serie de ensayos no destructivos realizados ya que ha sido objeto de numerosas investigaciones en los últimos años. El modelo es desarrollado anteriormente por varios autores (Rodriguez, 2014, Afonso, 2016).

3.2 Modelación del puente objeto de estudio

Para la confección del modelo el puente objeto de estudio se disponen de modelos mecánicos de la superestructura del mismo, elaborados previamente en SAP2000v16. Este proceso conlleva la declaración de los materiales, uniones, masas, estados de cargas, apoyos, etc., según los pasos llevados a cabo en el epígrafe 2.3.2. El puente queda confeccionado de forma general según (Foto 3.2 y 3.3):

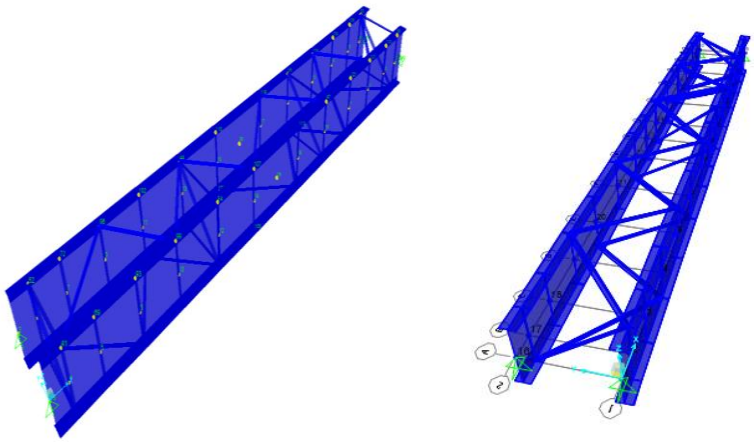


Foto 3.2: *Modelo de la superestructura del Puente km.9.578 de Línea Cenizas en SAP2000v16.*
Fuente: Elaboración Propia.

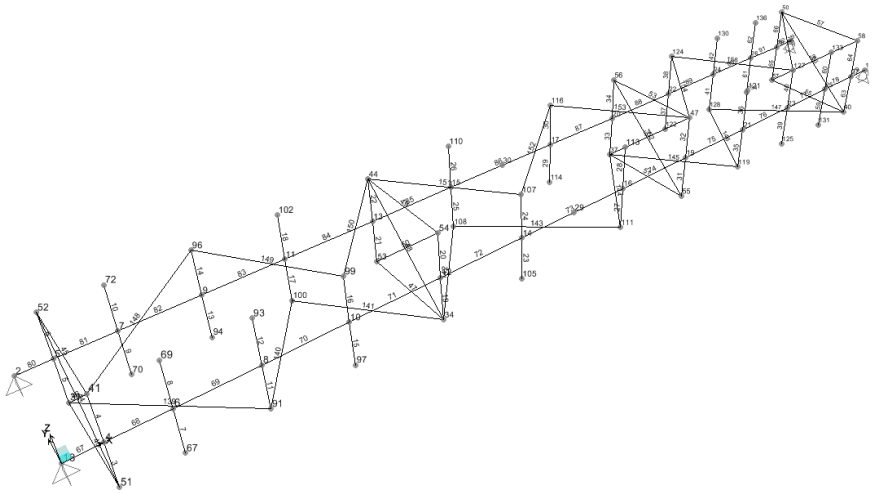


Foto 3.3: *Nudos y elementos del Modelo de la superestructura del Puente km.9.578 de Línea Cenizas en SAP2000v16. Fuente: Elaboración Propia.*

Como características generales se define que el puente es de acero, posee una única luz de 18.4m según su modelación en SAP2000, con una distancia entre ejes de 2.14m, con carriles tipo P-50, traviesa de madera dura mejicana y pino soviético para un total de 64u y estribos y pilas de muros de cantos. Este modelo no posee los elementos de madera, pero si los elementos de acero en su modelación. Dicho modelo es el que posee extensión *.sdb.

Para el análisis del puente a partir del algoritmo desarrollado se divide el mismo en 7 secciones, siendo estos los 7 tramos a calibrar (Foto 3.4):

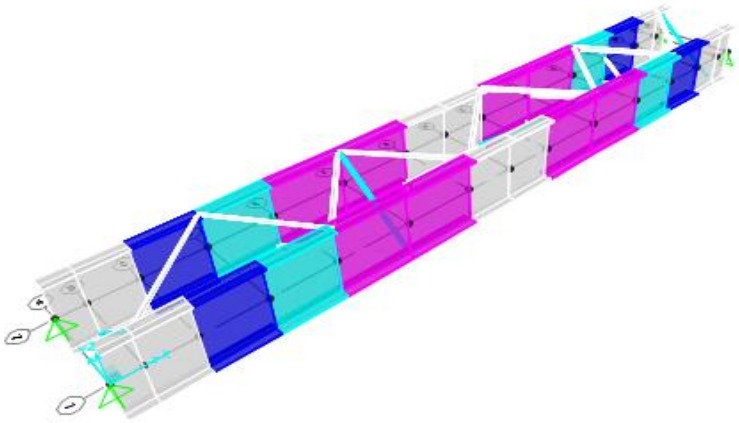


Foto 3.4: *Secciones del Modelo de la superestructura del Puente km.9.578 de Línea Cenizas en SAP2000v16. Fuente: Elaboración Propia.*

Esto permite obtener una mejor respuesta dinámica de la estructura, logrando hacer las iteraciones para cada uno de estos tramos y así arrojar mejores resultados en la

actualización del modelo. Para el uso del modelo creado en SAP2000v16 en la calibración fue necesario establecer una diferencia entre el modelo de referencia y el modelo actualizado.

El modelo de referencia no es más que aquel guardado como Ref_Model el cual posee una reducción de inercia de 0.35 y de 0.8 en el área para una sección creada con el nombre SEC5RED, el mismo tiene como objetivo simular un daño (daño inducido) en una de las secciones analizadas. Esto se hace ya que no existen parámetros dinámicos medidos obtenidos de mediciones experimentales conllevando a inducir daños en la estructura analizada para poder llevar a cabo el proceso de calibración. Mientras que Upd_Model es aquel que no posee daño artificial en ninguna de las secciones analizadas.

Concluida la creación del modelo fue creada una carpeta con el nombre *J_off* donde se incluye lo siguiente:

d) Carpeta con el modelo del puente en SAP2000 que posee extensión *.sdb con nombre: Ref_Model (contiene daño).
e) Carpeta con el modelo del puente en SAP2000 que posee extensión *.sdb con nombre: Upd_Model (sin daño).

3.3 Declaración de los parámetros que intervienen en la optimización

La selección de los parámetros y los componentes del problema de optimización para la calibración del modelo estructural del puente analizado conlleva un análisis adecuado, el cual permite obtener resultados satisfactorios. Estos parámetros están comprendidos desde la ubicación de los sensores con el respectivo análisis de sensibilidad hasta las variables de estado, diseño, restricciones y función objetivo.

3.3.1 Localización de los sensores

Para llevar a cabo un adecuado estudio de sensibilidad hay que tener en cuenta los sensores. En nuestro caso no se cuenta con mediciones reales del puente objeto de estudio, lo que de existir hubiera aportado las formas y frecuencias modales que servirían como variables de estado en el proceso de optimización. En esta investigación la localización de los sensores se hace a partir de los resultados que se quieran obtener.

Este proceso es conocido como Procesamiento de Señales (*Signal Processing*), donde las mediciones de los sensores se recogen en función del tiempo para luego a partir de transformadas de Fourier llevarlas al domino de las frecuencias y así ser utilizadas con diferentes fines. Al no contar con un estudio de mediciones es de vital importancia declarar

correctamente la posición de los sensores ya que el objetivo es alcanzar con mayor precisión las características dinámicas del puente analizado en cuanto a las frecuencias y las formas modales. Al colocar la mayor cantidad de sensores posibles sería más información de entrada para la optimización lo que daría mayor precisión en los resultados.

Los sensores corresponden a puntos en el modelo en SAP2000v16 de los cuales se extraen las coordenadas modales en cada iteración hasta que se calibre. En este caso son (1,2,3,4,5,6,7, 8…22,23,24,25,26,27,28,29,30), ver foto 3.5. También el objetivo es tratar de colocar la mayor cantidad de sensores posibles debido a que sería más información de entrada para la optimización lo que daría mayor precisión en los resultados teniendo en cuenta los distintos movimientos posibles (0.01, 0.02, 0.03, 0.04, 0.05, 0.06) en la dirección en que serán medidos.

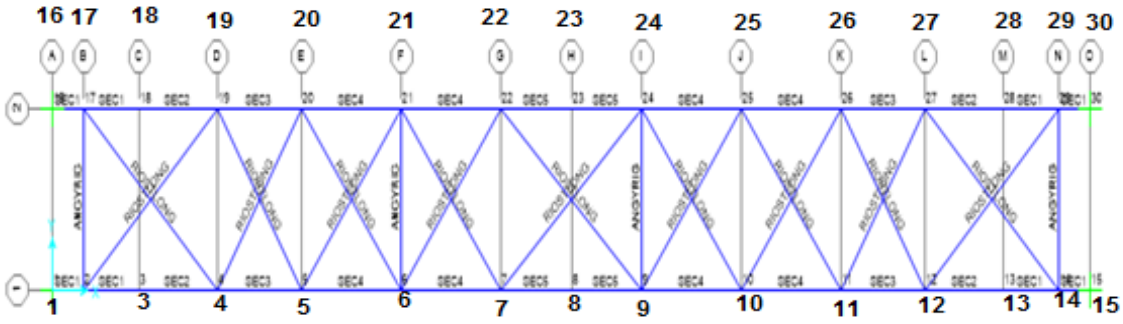


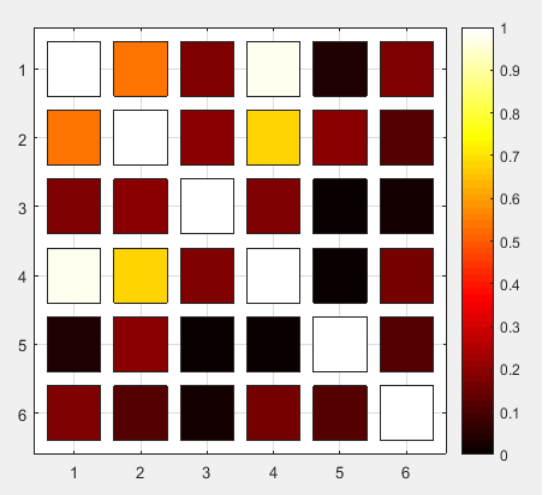
Foto 3.5: *Ubicación de los sensores en los puntos (1-15) al (16-30). Fuente: Elaboración Propia.*

En el análisis de sensibilidad se tiene en cuenta el uso de la matriz MAC, la misma posibilita la validación del modelo y la calibración de este. Para esto se hace uso del fichero que tiene en cuenta el análisis de esta matriz ***sensitivity_analysis***, el cual tiene como finalidad lograr mediante la confección de un gráfico que en la diagonal principal de la misma se logre los valores más cercanos posible a 1 (en este caso se identifica con el color blanco en la diagonal principal), lo que implica una buena correlación entre las formas modales del modelo numérico y las formas modales de la estructura real. Ver foto 3.6.

Definidos en este *script* las siguientes posiciones de los sensores:

DOF_sensors= [(1:15, 16:30).'+0.01; (1:15, 16:30).'+0.02; (1:15, 16:30).'+0.03];

Obteniéndose la siguiente matriz MAC:



1.0000	0.5341	0.1809	0.9698	0.0465	0.1872
0.5341	1	0.1919	0.6849	0.1923	0.1244
0.1809	0.1919	1	0.1748	6.0659e-05	0.0183
0.9698	0.6849	0.1748	1	0.0079	0.1647
0.0465	0.1923	6.0659e-05	0.0079	1	0.1114
0.1872	0.1244	0.0183	0.1647	0.1114	1

Foto 3.6: *Matriz MAC. Fuente: Elaboración Propia.*

Este resultado implica que se acepta la posición de los sensores propuesta debido a que la diagonal principal tiene valores de 1 y los restantes valores se encuentran dentro de los parámetros establecidos.

3.3.2 Variables de diseño

Para este caso de estudio las variables independientes o de diseño son aquellas que son iteradas en orden para alcanzar los valores óptimos. Para esta investigación se declara una variable inicial $X\_ref_{(1...7)} = E_1 \ldots E_7$ (módulo de deformación), siendo la única variable que se calibra para los 7 tramos del puente analizado. Estas poseen valores únicos e iguales para todos los tramos (200 GPa).

3.3.3 Variables de estado

Las variables de estado no son más que las frecuencias naturales de los 6 primeros modos de oscilación flexionales simétricos del puente y las coordenadas medidas en los puntos que corresponden a los sensores ubicados. Estos parámetros son dependientes pues dependen de los valores que toma cada variable de diseño para cada una de las iteraciones.

3.3.4 Función Objetivo

La función objetivo como se ha ido mencionando en epígrafes anteriores es la función del vector residual entre los resultados obtenidos en las mediciones y los obtenidos en el modelo.

En este caso se tienen en cuenta los «*weighing factors*» o factores de peso α y β los cuales reflejan el peso señalado (normalizado) para los valores propios y los residuos de las formas modales. Los valores asignados fueron de 1 para ambos casos.

3.3.5 Restricciones

Las restricciones son los límites inferior y superior que deben cumplir las variables de diseño. Estos límites deben comprender en su intervalo al valor real al que la calibración debe arribar, sino el algoritmo tardará mucho o los valores que encontrará serían irreales por completo. En nuestro caso se definen:

- E_0 = 2.1 * 1e8 KPa
- $E_{mín}$ = 0.35 * E_0
- $E_{máx}$ = 1.50 * E_0
- Lb = $E_{mín}$ = 73.5 GPa
- Ub = $E_{máx}$ = 315 GPa

Los valores asignados son cercanos a la inicial para que el algoritmo de optimización no se hiciera muy extenso o que no pudiera encontrar la convergencia.

Para el caso del uso del algoritmo que utiliza el *Levenberg-Marquardt Method* se definen límites a las variables de diseño de -infinito y +infinito para un E_0 = 2.1 * 1e8 KPa, debido a que este algoritmo de optimización no permite que se le definan restricciones al problema.

3.4 Obtención de los parámetros modales del modelo. Análisis de resultados

Al ejecutar el MATLAB se procede a la creación de los ficheros que intervienen en la calibración (***data_bridge, eigfem_bridge, main_bridge, sensitivity_analysis, upd_bridge***) para el uso de la matriz Jacobiana en *off* y (***data_bridge, run_SAP2k, main_bridge, sensitivity_analysis, upd_bridge, asmkm_sap2k.m***) para el caso en *on*, en los que se incorporaron como parámetros que intervienen las mencionadas anteriormente en el epígrafe 3.3.

3.4.1 Resultados: Caso Jacobian *off*

3.4.1.1 Caso_1: *Trust-Region-Reflective Least Squares Algorithm*

En la aplicación de la matriz Jacobiana en *off* para la optimización se tuvo en cuenta los parámetros declarados en el epígrafe anterior, donde se calibran 7 módulos de

deformación para los 7 tramos del puente objeto de estudio y se utiliza el *Trust-Region-Reflective Least Squares Algorithm* como método de optimización.

La calibración terminó en **1h y 35min** con un total de **87 iteraciones**, deteniéndose ya que *lsqnonlin* excede el límite de evaluación de las opciones de la función (*MaxFunEvals* = 700**)** En el siguiente Gráfico 3.1 se muestra los residuos para cada iteración.

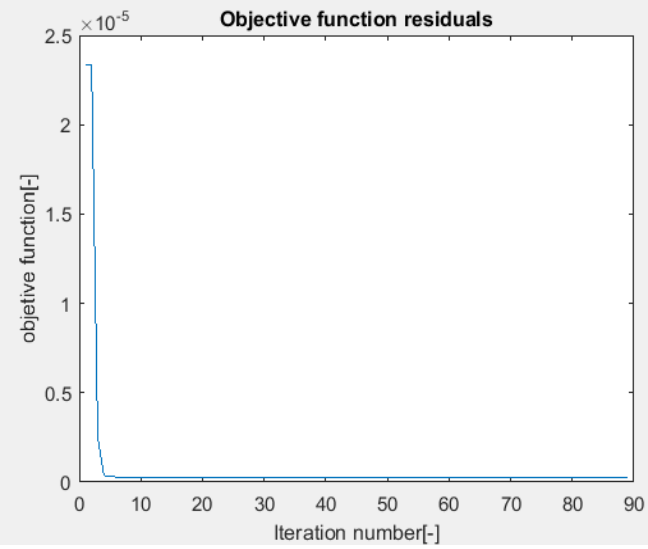


Gráfico 3.1: *Residuos de la función objetivo para cada iteración*

En la siguiente Tabla 3.1 se resumen los valores de las frecuencias propias de oscilación para los 6 primeros modos, los cuales fueron los tomados en cuenta a la hora de la calibración.

Modos	***Frecuencias propias de oscilación***			***Error en las frecuencias (%)***		***Valores MAC (%)***	
	Valor Expert.	***Modelo Inicial***	***Modelo Calib.***	***Modelo Inicial***	***Modelo Calib.***	***Modelo Inicial***	***Modelo Calib.***
1	2.69	2.76	2.64	2.61	1.88	99.99867	99.98771
2	7.25	7.44	7.16	2.53	1.35	99.99188	99.95765
3	7.80	8.09	7.86	3.73	0.84	99.98234	99.96702
4	12.48	13.34	12.66	6.94	1.48	99.97867	99.3382
5	13.14	13.88	13.36	5.59	1.68	99.70487	99.77253
6	15.81	16.55	15.90	4.65	0.57	98.53193	99.89671

Tabla 3.1*: Frecuencias propias de oscilación, errores en las mismas y valores de la matriz MAC para los valores experimentales y los modelos inicial y calibrado*

En el Gráfico 3.2 se muestra los valores de las frecuencias para cada modo analizado para el modelo inicial, calibrado y las frecuencias de mediciones experimentales.

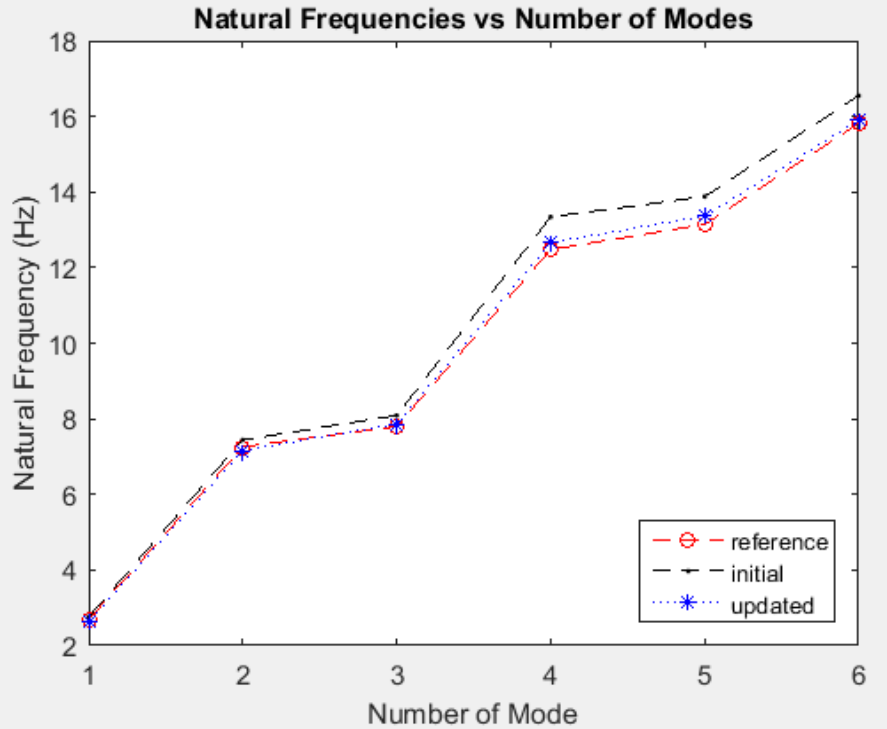
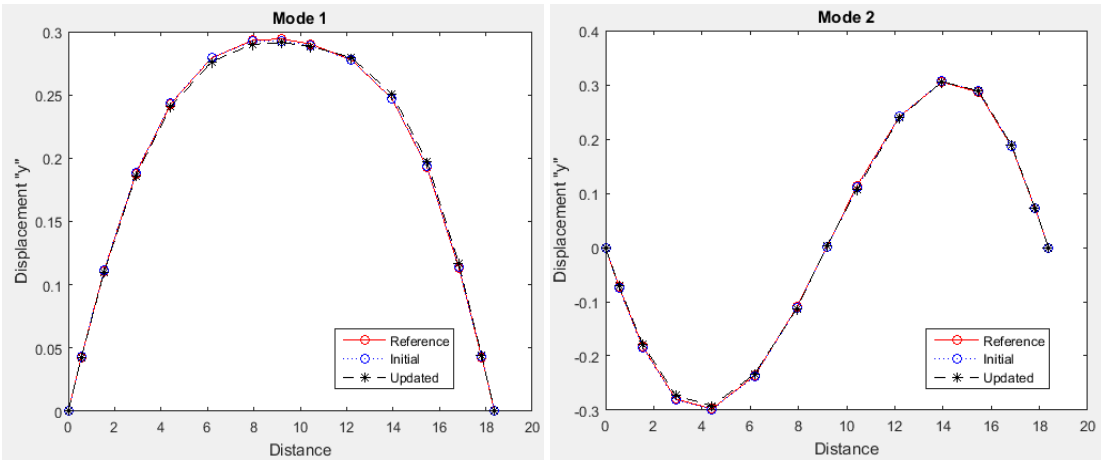


***Gráfico 3.2**: Frecuencias propias de oscilación para los 6 primeros modos correspondientes a los valores experimentales y a los modelos inicial y calibrado.*

El Gráfico 3.3 muestra los valores de los desplazamientos modales para las dos direcciones en que se desplaza el puente «z», «y» y «torsional-z» para los valores de referencia, del modelo inicial y del modelo calibrado. Los nodos que se tienen en cuenta son los definidos anteriormente (1:15-16:30).



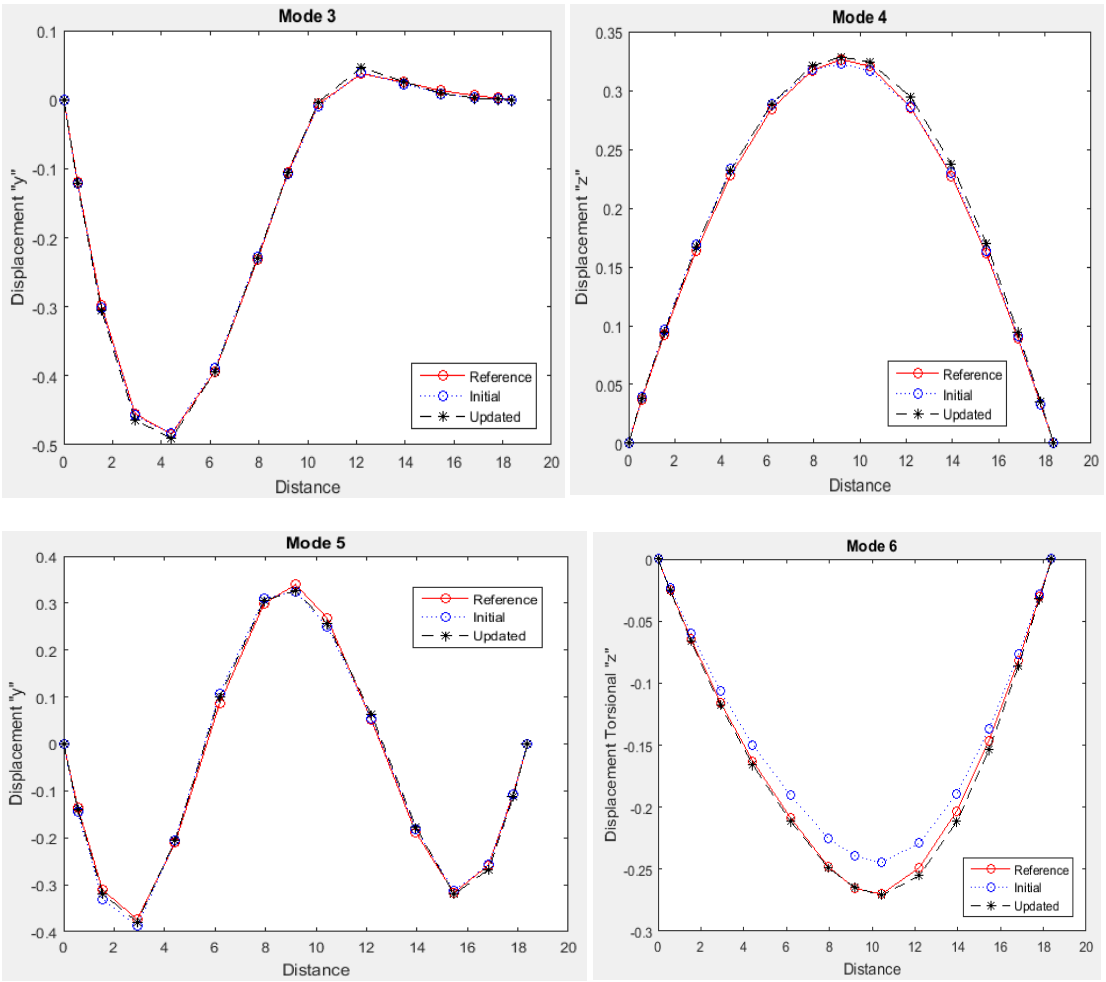


Gráfico 3.3: *Valores de las coordenadas modales para cada modo de los valores experimentales y los modelos inicial y calibrado*

Teniendo en cuenta que los valores experimentales o de referencia son calculados a partir de un modelo dañado del puente objeto de estudio (reducción de inercia de 0.35 y de área de 0.8 aplicado en Ref_Model), se presentan en la Tabla 3.2 los valores del módulo de deformación que se incorporaron como valores iniciales y los calculados al terminar la calibración.

Modelo dañado	***Modelo no dañado***	
Valores de E de referencia (x_ref)	***Valores de E Iniciales (E_0)***	***Valores de E calculados (E_upd)***
200000000	210000000	197933220.81
200000000	210000000	179386283.60
200000000	210000000	198403570.58
200000000	210000000	182145542.59
200000000	210000000	202057650.06
200000000	210000000	257096050.94
200000000	210000000	210000000.00

Tabla 3.2*: Valores de E de referencia y los obtenidos al finalizar la calibración.*

3.4.1.2 Caso_2: *Levenberg-Marquardt Method*

En la aplicación de la matriz Jacobiana en *off* para el caso del uso de *Levenberg-Marquardt Method* para la optimización se tuvo en cuenta los mismos parámetros declarados en el proceso anterior, donde se calibran 7 módulos de deformación para los 7 tramos del puente objeto de estudio. La calibración terminó en **18min** con un total de **13 iteraciones**, la optimización se detiene ya que la norma relativa del paso es menor a las opciones. En el siguiente Gráfico 3.4 se muestra los residuos para cada iteración.

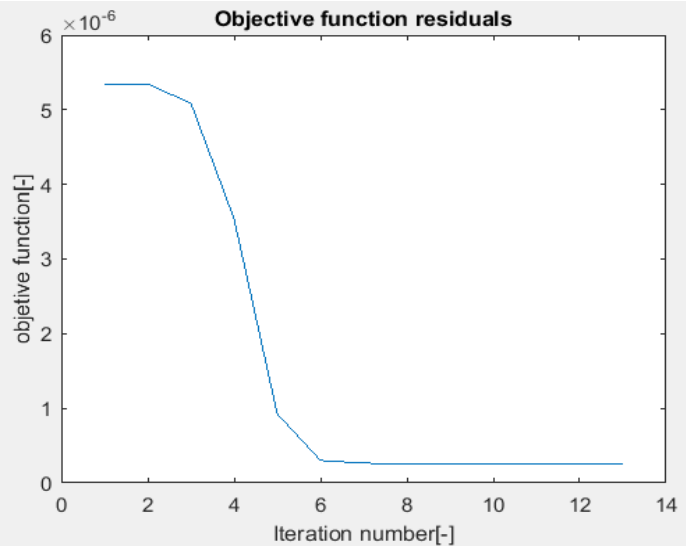


Gráfico 3.4: *Residuos de la función objetivo para cada iteración*

En la siguiente Tabla 3.3 se resumen los valores de las frecuencias propias de oscilación para los 6 primeros modos, los cuales fueron los tomados en cuenta a la hora de la calibración.

Modos	***Frecuencias propias de oscilación***			***Error en las frecuencias (%)***		***Valores MAC (%)***	
	Valor Expert.	***Modelo Inicial***	***Modelo Calib.***	***Modelo Inicial***	***Modelo Calib.***	***Modelo Inicial***	***Modelo Calib.***
1	2.76	2.76	2.70	0.16	1.88	99.99873	99.98746
2	7.43	7.44	7.34	0.09	1.30	99.99479	99.95719
3	7.99	8.09	8.05	1.28	0.93	99.98764	99.96397
4	12.78	13.34	12.96	4.40	1.44	99.98088	99.93083
5	13.46	13.88	13.69	3.11	1.70	99.72075	99.77163
6	16.19	16.55	16.29	2.19	0.60	98.58195	99.89331

Tabla 3.3: *Frecuencias propias de oscilación, errores en las mismas y valores de la matriz MAC para los valores experimentales y los modelos inicial y calibrado*

En el Gráfico 3.5 se muestra los valores de las frecuencias para cada modo analizado para el modelo inicial, calibrado y las frecuencias de mediciones experimentales.

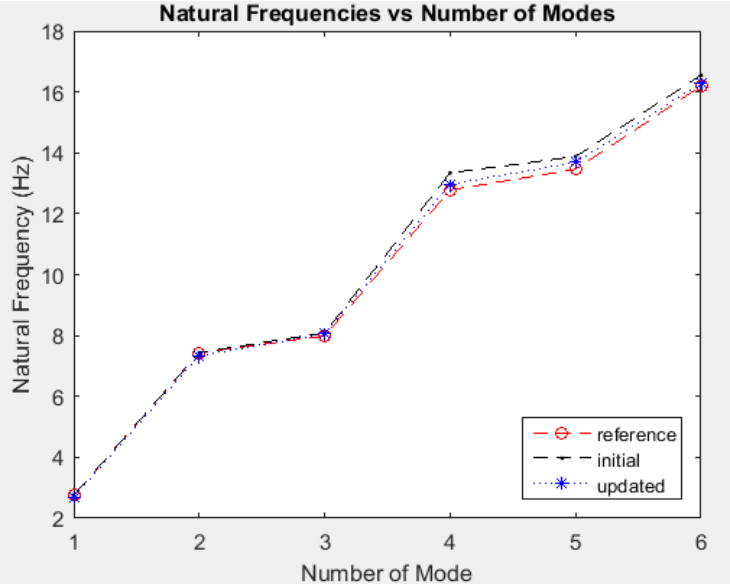
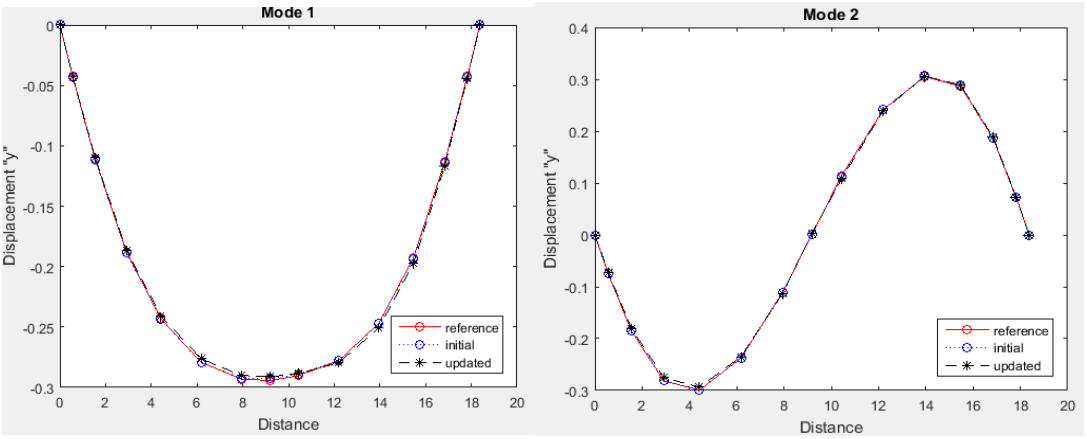


***Gráfico 3.5**: Frecuencias propias de oscilación para los 6 primeros modos correspondientes a los valores experimentales y a los modelos inicial y calibrado.*

El Gráfico 3.6 muestra los valores de los desplazamientos modales para las dos direcciones en que se desplaza el puente «z», «y» y «torsional-z» para los valores de referencia, del modelo inicial y del modelo calibrado. Los nodos que se tienen en cuenta son los definidos anteriormente (1:15-16:30).



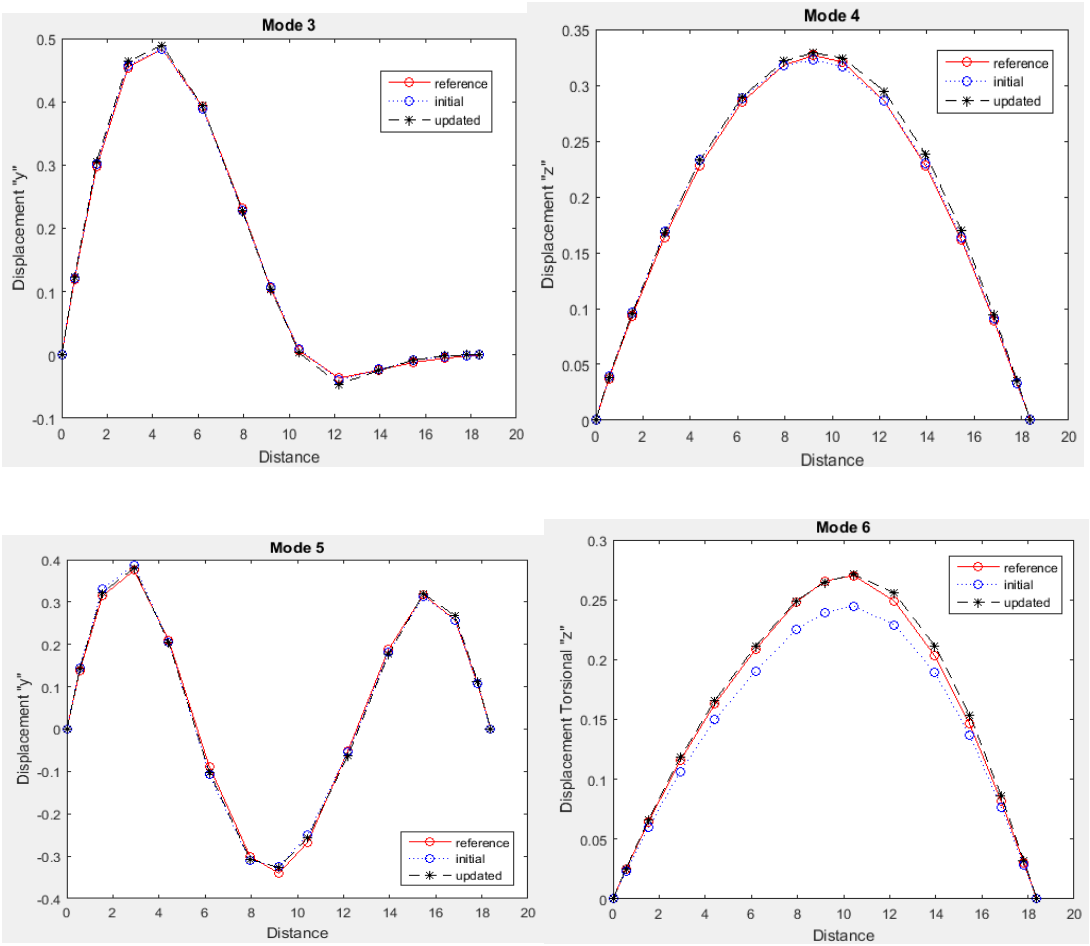


***Gráfico 3.6**: Valores de las coordenadas modales para cada modo de los valores experimentales y los modelos inicial y calibrado.*

Teniendo en cuenta que los valores experimentales o de referencia son calculados a partir de un modelo dañado del puente objeto de estudio (Ref_Model), se presentan en la Tabla 3.4 los valores del módulo de deformación que se incorporaron como valores iniciales y los calculados al terminar la calibración.

Modelo dañado	***Modelo no dañado***	
Valores de E de referencia (x_ref)	***Valores de E Iniciales (E_0)***	***Valores de E calculados (E_upd)***
200000000	210000000	207148604.72
200000000	210000000	188350197.25
200000000	210000000	208288822.95
200000000	210000000	191293054.55
200000000	210000000	212382832.74
200000000	210000000	278669692.46
200000000	210000000	266850750.00

***Tabla 3.4**: Valores de E de referencia y los obtenidos al finalizar la calibración.*

3.4.2 Resultados: Caso Jacobian *on*

En la aplicación de la matriz Jacobiana en *on* para la optimización se tuvo en cuenta los mismos parámetros declarados en el epígrafe anterior y los utilizados para el Jacobiano en *off*, para así ver la diferencia entre los resultados de ambos casos.

3.4.2.1 Caso_1: *Trust-Region-Reflective Least Squares Algorithm*

Para el uso del *Trust-Region-Reflective Least Squares Algorithm* en este caso también se calibran 7 módulos de deformación para los 7 tramos del puente objeto de estudio. La calibración terminó en **26min** con un total de **12 iteraciones**, deteniéndose ya que el tamaño del paso es mayor a la tolerancia establecida. En el siguiente Gráfico 3.7 se muestra los residuos para cada iteración. Los residuos poseen un valor único de 2.33209e-05 para todas las iteraciones.

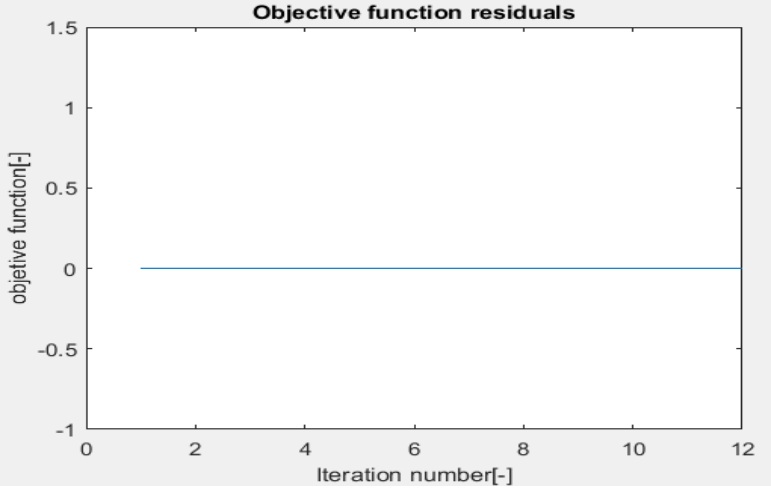


***Gráfico 3.7**: Residuos de la función objetivo para cada iteración*

En la siguiente Tabla 3.5 se resumen los valores de las frecuencias propias de oscilación para los 6 primeros modos, los cuales fueron los tomados en cuenta a la hora de la calibración.

Modos	***Frecuencias propias de oscilación***			***Error en las frecuencias (%)***		***Valores MAC (%)***	
	Valor Expert.	***Modelo Inicial***	***Modelo Calib.***	***Modelo Inicial***	***Modelo Calib.***	***Modelo Inicial***	***Modelo Calib.***
1	2.69	2.76	2.76	2.61	2.61	-	-
2	7.25	7.44	7.44	2.53	2.53	-	-
3	7.80	8.09	8.09	3.73	3.73	-	-

4	12.48	13.34	13.34	6.94	6.94	-	-
5	13.14	13.88	13.88	5.59	5.59	-	-
6	15.81	16.55	16.55	4.65	4.65	-	-

***Tabla 3.5**: Frecuencias propias de oscilación, errores en las mismas y valores de la matriz MAC para los valores experimentales y los modelos inicial y calibrado*

En el Gráfico 3.8 se muestra los valores de las frecuencias para cada modo analizado para el modelo inicial, calibrado y las frecuencias de mediciones experimentales.

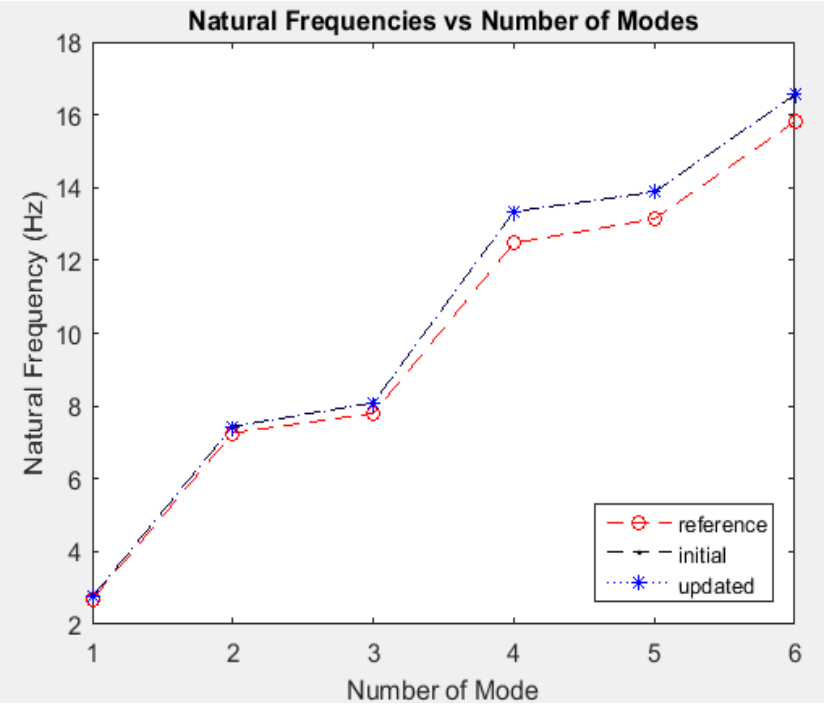


***Gráfico 3.8**: Frecuencias propias de oscilación para los 5 primeros modos correspondientes a los valores experimentales y a los modelos inicial y calibrado.*

El Gráfico 3.9 muestra los valores de los desplazamientos modales para las dos direcciones en que se desplaza el puente «z», «y» y «torsional-z» para los valores de referencia, del modelo inicial y del modelo calibrado. Los nodos que se tienen en cuenta son los definidos anteriormente (1:15-16:30).

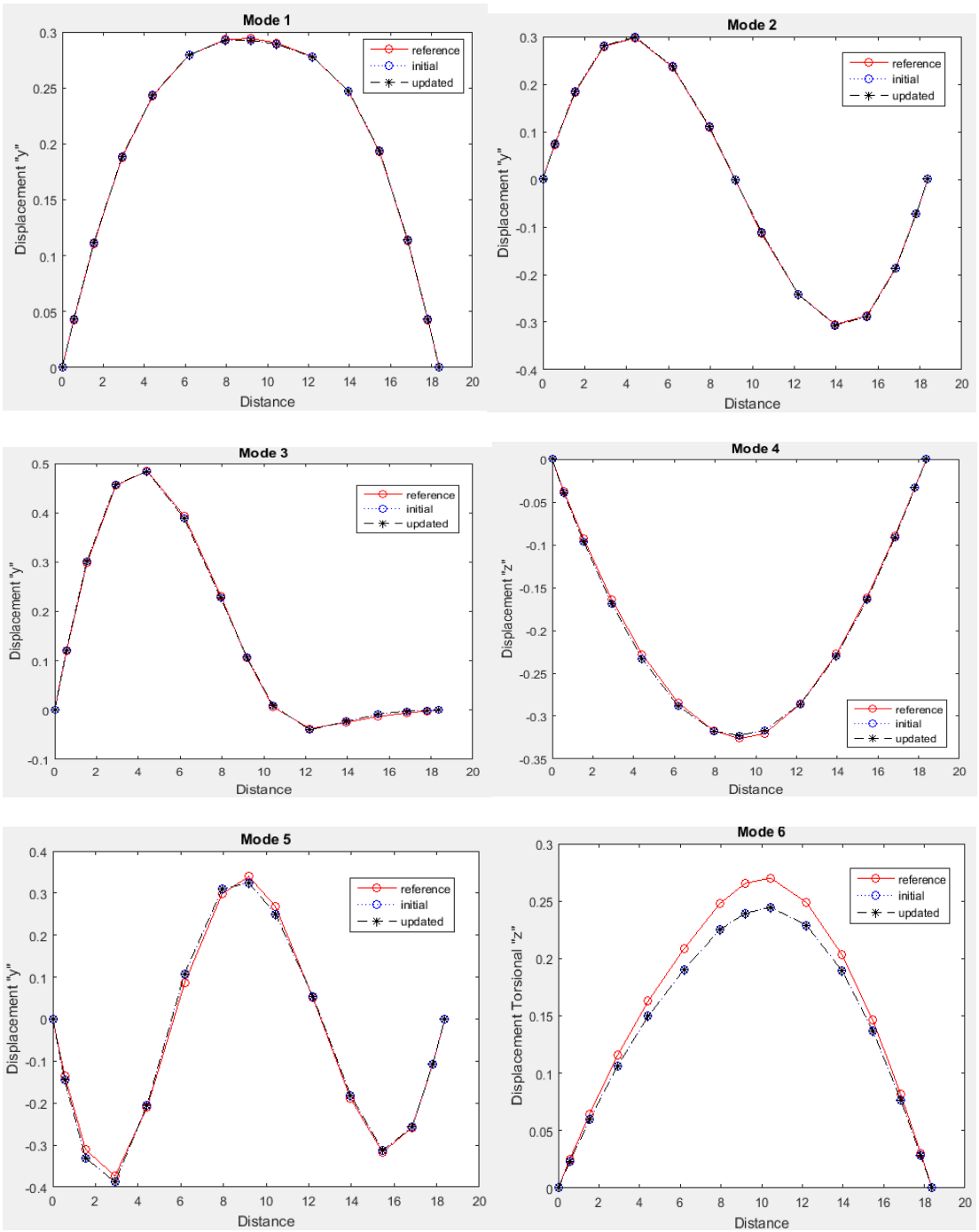


***Gráfico 3.9**: Valores de las coordenadas modales para cada modo de los valores experimentales y los modelos inicial y calibrado*

Teniendo en cuenta que los valores experimentales o de referencia son calculados a partir de un modelo dañado del puente objeto de estudio (Ref_Model), se presentan en la Tabla 3.6 los valores del módulo de deformación que se incorporaron como valores iniciales y los calculados al terminar la calibración.

Modelo dañado	***Modelo no dañado***	
Valores de E de referencia (x_ref)	***Valores de E Iniciales (E_0)***	***Valores de E calculados (E_upd)***
200000000	210000000	210000000
200000000	210000000	210000000
200000000	210000000	210000000
200000000	210000000	210000000
200000000	210000000	210000000
200000000	210000000	210000000
200000000	210000000	210000000

***Tabla 3.6**: Valores de E de referencia y los obtenidos al finalizar la calibración.*

3.4.2.2 Caso_2: *Levenberg-Marquardt Method*

En la aplicación de la matriz Jacobiana en *on* para la optimización e implementando el *Levenberg-Marquardt Method* se tuvo en cuenta los mismos parámetros ya declarados. En este proceso se calibran también 7 módulos de deformación para los 7 tramos del puente objeto de estudio. La calibración en este caso arroja residuos que poseen un valor único de 2.33209e-05, para un proceso que dura solo **12min**. En este caso_2 no se exponen los resultados como en los casos anteriores pues los resultados referentes al trabajo con las frecuencias propias, los 6 modos analizados y las E_upd expuestos en gráficos y tablas son los mismos obtenidos para el caso de la matriz Jacobiana en *on* y el uso del *Trust-Region-Reflective Least Squares Algorithm*.

3.4.3 Análisis de los resultados

Para llevar a cabo el análisis de los resultados hay que tener en cuenta que todas las calibraciones fueron realizadas en un mismo ordenador, este ordenador no cuenta con buenas características (Procesador: *AMD Athlon™ II X2 255 Processor* 3.10GHz, 2GB de RAM con sistema operativo de 64bits) las cuales pueden influir en el tiempo empleado para llevar acabo todo el procedimiento.

El análisis de los residuales en la función objetivo teniendo en cuenta los gráficos (3.1 y 3.4) para el Jacobiano en *off* muestra una disminución a medida que aumentan las iteraciones. Esto demuestra que el algoritmo de optimización implementado fue encontrando la convergencia, pero al alcanzar un punto donde se mantiene constante comienza a tener pérdidas de tiempo o iteraciones innecesarias, sin importar el método de optimización empleado. En los dos casos de estudio aparecen residuos pequeños, pero

no llegan a 1,0e-16 (valor asignado para obtener una mejor optimización) lo que demuestra poca precisión en la calibración, aunque se obtienen valores aceptables en el rango de 1,0e-7. En ambos métodos empleados ocurre lo mismo.

Analizando el Gráfico (3.7) para el Jacobiano en *on*, se observa como los residuos se mantienen constantes para las iteraciones, lo que demuestra un resultado poco satisfactorio, pues no se logra la calibración.

Al analizar las tablas (3.1 y 3.3) para el caso en *off,* se observa que existe una diferencia entre las frecuencias de oscilación iniciales y actualizadas, siendo similares para los valores de referencia y actualizados, obteniéndose como resultado errores inferiores al 2% en estas diferencias. Estos valores se consideran aceptables, pero demuestran como ambos métodos no logran alcanzar los valores óptimos para la correspondencia entre la respuesta modelada y la medida, pues no se obtiene 100% de calibración. Mientras que para la Tabla (3.5) en el caso en *on* se observa que no existe diferencia entre los valores iniciales y los actualizados, pues no se realiza la calibración, reafirmando lo expuesto para los residuos.

Los valores MAC iniciales para el caso en *off* muestran valores muy cercanos a 100, mientras que, al finalizar, estos valores mantienen su cercanía a 100 pero no llega a este, mostrando que existe una posible coincidencia entre los modos experimentales y los obtenidos al final de la calibración. Para *on* no ocurren cambios, esta matriz se indefine pues no se realiza la calibración (no existe una coincidencia entre los modos experimentales y los obtenidos).

Las coordenadas modales en general para los 6 modos analizados en los casos en *off* (Gráficos 3.3 y 3.6) para el modelo inicial difieren de los valores experimentales y al terminar la calibración se logra establecer una coincidencia casi exacta en los 6 modos analizados entre los valores de referencia y actualizados, demostrando que se encontró un punto óptimo, manifestando efectividad en el proceso de forma general. Para el caso en *on* (Gráfico 3.9) coinciden los valores iniciales con los calibrados siendo diferentes estos de los experimentales, no lográndose así la calibración.

En la obtención de E_upd se muestra en todos los casos, Tablas (3.2, 3.4 y 3.6), que se obtienen los valores que el programa mediante la optimización cree correctos para el juego de datos con el que trabaja para los 7 tramos analizados. Analizando la correspondencia entre los valores modelados y los obtenidos se puede decir que aparece grandes diferencias entre ambos.

En cuanto a los métodos empleados para la optimización: *Trust-Region-Reflective Least Squares Algorithm* y *Levenberg-Marquardt Method,* se puede decir que el primero ocupa un mayor tiempo de calibración, tiende a no encontrar el mínimo correctamente o llegar a un punto donde los residuos se mantienen constantes y la optimización se detiene debido a que la distancia del paso es muy pequeña por debajo de la tolerancia permitida u ocurre un exceso de iteraciones. Mientras que, en el segundo, el proceso es más rápido, pero igualmente en un momento se detiene ya que la suma relativa de cuadrados supera el límite de las opciones, esto puede traer consigo quedarse por debajo de buenos resultados en la calibración, pero aun así es superior en velocidad de convergencia. En cuanto a la salida de los resultados se puede decir que ambos devuelven valores muy similares, pero no se logra un 100% de calibración.

En la siguiente Tabla 3.6 se muestra un resumen de los resultados obtenidos para cada caso analizado.

Elementos de comparación	**Usos de la Matriz Jacobiana**			
	J_off		**J_on**	
	Caso_1	**Caso_2**	**Caso_1**	**Caso_2**
Tiempo empleado	1h y 35min	18min	26min	12min
Residuos	1.0e-07	1.0e-07	2.33209e-05 (valor único)	2.33209e-05 (valor único)
Iteraciones	87	13	12	1
Errores en las Frecuencias (promedio)	1.3%	1.3%	No calibra	No calibra
Calibración	Sí	Sí	No	No

Tabla 3.6: Resumen de los resultados.

Caso_1: *Trust-Region-Reflective Least Squares Algorithm*

Caso_2: *Levenberg-Marquardt Method*

3.5 Conclusiones parciales del capítulo

Implementado el procedimiento para la calibración de modelos estructurales utilizando la interface OAPI SAP2000-MATLAB que incluye la matriz Jacobiana (*on-off*) se llega a las siguientes conclusiones:

- El proceso de modelación fue llevado a cabo desde SAP2000v16, creándose dos modelos para el trabajo: el primero con daño inducido para trabajar con los valores experimentales y uno actualizado para llevar a cabo la calibración, ambos con la división en 7 secciones.
- La inexistencia de parámetros dinámicos medidos o experimentales obtenidos de mediciones experimentales conlleva a la inducción de daños en la estrutura analizada para poder llevar a cabo el proceso de calibración.
- El análisis de sensibilidad teniendo en cuenta el uso de la matriz MAC, posibilita la validación del modelo y la calibración de este a partir del uso de *sensitivity_analysis,* aunque no se obtengan resultados satisfactorios en general.
- La calibración realizada se implementó para la actualización de 7 módulos de deformación (E) en 7 tramos del puente objeto de estudio, en los cuales se hizo un análisis de los residuos de la función objetivo, el trabajo con las frecuencias de referencia, iniciales y actualizadas con sus errores correspondientes, así como el trabajo con los 6 modos analizados.
- La implementación de la matriz Jacobiana en *off* presentó mejores resultados ante el de su uso en *on*, demostrando la existencia de fallas en el procedimiento creado, el cual no logra su objetivo general: calibrar.
- El procedimiento creado para el uso de la matriz Jacobiana en *on* no se recomienda para su uso, se debe trabajar en sus deficiencias, dadas principalmente por el trabajo ineficaz con las matrices y la base matemática del Método de Nelson.
- Se aplicaron como métodos de optimización para la función *lsqnonlin*: el *Trust-Region-Reflective Least Squares Algorithm* y el *Levenberg-Marquardt Method* los cuales presentaron diferencias en cuanto a velocidad de convergencia y similitud a la hora de la obtención de los valores óptimos de calibración.
- Las posibles fuentes de error están dadas por la no existencia de mediciones reales, lo que implica una inducción de daño por el usuario, trayendo consigo reducciones de las propiedades de los elementos que no sean las adecuadas. Se debe tener en cuenta los rangos de las restricciones, localización de los sensores, la base matemática de los métodos a utilizar y el sistema de referencia (x, y, z) utilizado.

CONCLUSIONES

Acerca de la revisión del estado del conocimiento de la calibración de modelos estructurales, también conocido como *Model Updating*, se puede decir que existen disimiles métodos y procedimientos para llevarlos a cabo, los mismos se basan en el uso de softwares como MATLAB, SAP2000, StaBIL-2.0, ANSYS, etc., que interactúan para la modelación y optimización, utilizando un amplio trabajo de programación de algoritmos y una vasta base matemática para cada método.

La interface OAPI SAP2000 garantiza la interacción entre SAP2000 y MATLAB, permitiendo que la optimización en la calibración de los modelos mecánicos de puentes sea desarrollada minimizando los residuos entre la respuesta modelada y la medida.

El uso del análisis de sensibilidad con la matriz MAC admite la validación del modelo y la calibración de este, lo que permite obtener una correlación perfecta entre las formas modales del modelo numérico y las formas modales de la estructura real.

La matriz Jacobiana está formada por las derivadas parciales de primer orden de la función objetivo que se va a minimizar, esta se encuentra dentro de la función *lsqnonlin* y su uso está condicionado por las opciones *(off-on)*, lo que establece su uso o no dentro del procedimiento creado. La matriz Jacobiana va buscando los valores donde el gradiente de la función objetivo cambia su signo, esto indica que el punto óptimo está próximo al analizado.

El procedimiento para el uso de la matriz Jacobiana en *on* presentó grandes deficiencias para calibrar, objetivo que no fue cumplido debido a la complejidad presentada principalmente por la base matemática que conlleva su uso, además hay que tener en cuenta: a) errores dados por la no existencia de mediciones reales, lo que implica una inducción de daño por el usuario, trayendo consigo reducciones de las propiedades de los elementos que no sean las adecuadas, b) los rangos de las restricciones, c) localización de los sensores, d) la base matemática de los métodos a utilizar (Método de Nelson y función objetivo) y f) el sistema de referencia (x, y, z) utilizado, debido a que este es diferente en los programas empleados (SAP2000 y StaBIL-2.0).

El caso del uso de la matriz Jacobiana en *off* realiza la calibración, pero los residuos quedan por debajo de las tolerancias establecidas, valores que determinarían la existencia de menores errores.

RECOMENDACIONES

1. Dedicar esta investigación no solo a puentes sino también a otros tipos de estructuras.
2. Estudiar y perfeccionar el uso de la matriz Jacobiana en *on* dentro del procedimiento creado para lograr la calibración, esto incluye tanto la base matemática del método así como los métodos de optimización adecuados a aplicar en cada caso.
3. Recurrir además a otros tipos de programas con las interacciones posibles entre ellos, como:
   - SAP2000-MATLAB-StaBIL-2.0
   - MATLAB-StaBIL-2.0
   - ANSYS-MATLAB
   - ANSYS con su propio toolbox de optimización
   - ABAQUS-MATLAB
   - SAP2000-Microsoft Excel
   - ETABS-MATLAB
4. Extender el procedimiento implementado a la calibración de otros parámetros como la reducción de las inercias en los perfiles, rigidez de las uniones, la resistencia del acero, etc.

REFERENCIAS BIBLIOGRÁFICAS

1. ADHIAKRI, S. 1999. Rates of Change of Eigenvalues and Eigenvectors in Damped Dynamic System. *AIAA Journal,* 39**,** 1452-1457.

2. AFONSO, Y. 2016. *Estimación de parámetros estructurales en puentes usando la OAPI SAP2000-MATLAB.* UCLV.

3. ALLEMANG, R. 2003. The Modal Assurance Criterion - Twenty Years of Use and Abuse. *SOUND AND VIBRATION* August**,** 14-21.

4. ANAYA, C. & BARAJAS, C. 2011. *Metodología para la detección de daños en estructuras metálicas empleando la técnica de análisis teórico-experimental.* Universidad Industrial de Santander.

5. ANCONA, A., SALGADO, R., ZAMORA, S. & MARTÍNEZ, F. 2011. Evaluación de métodos de detección de daños en estructuras mediante el uso de vibraciones. *XVIII Congreso Nacional de Ingeniería Sísmica*.

6. ANDERSEN, P. 2013. Experimental Modal Analysis and Operational Modal Analysis. Structural Vibration Solutions.

7. ANÓNIMO 2015. Análisis modal experimental.

8. ARAGÓN, Y. 2010. *Modelación estructural de un puente metálico de ferrocarril de viga, tablero superior.* UCLV.

9. BACUILIMA, F. L. 2011. *Técnicas de optimización para la identificación de parámetros mecánicos de estructuras.* Universidad de Cuenca.

10. BLUM, C. & ROLI, A. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys,* 35**,** 268–308.

11. CARRIÓN VIRAMONTES, F., LOZANO GUZMÁN, A., FABELA GALLEGOS, M. & VÁZQUEZ VEGA, D. 1999. Evaluación de puentes mediante el análisis de vibraciones. *Publicación Técnica No. 132.*

12. CLARO, A. 2015. *Métodos para la modelación y el análisis experimental de puentes frente a cargas dinámicas.*, UCLV.

13. COMPUTERS AND STRUCTURES, I. 2013. *Introductory Tutorial for SAP2000® Linear and Nonlinear Static and Dynamic Analysis and Design of Three-Dimensional Structures*.

14. CUJIA MEZA, Y. D. 2010. *Computación en paralelo aplicada a la optimización del diseño estructural: estado del arte.*

15. CHAGOYÉN, E. 2015. OAPI SAP 2000 MatLab. *Introducción OAPI SAP 2000 Matlab.* Facultad de Construcciones, Departamento de Ing Civil, UCLV.

16. DOOMS, D., JANSEN, M., DE ROECK, G., DEGRANDE, G., LOMBAERT, G., SCHEVENELS, M. & FRANCOIS, S. 2010. StaBIL: A FINITE ELEMENT TOOLBOX FOR MATLAB. *VERSION 2.0 USER'S GUIDE.* KU Leuven.

17. FOX, R. & KAPOO, M. 1968. Rates of Change of Eigenvalues and Eigenvectors. *AIAA Journal,* 6**,** 2426-2429.

18. FRISWELL, M. & ADHIKARI, S. 1976. Derivates of Complex eigenvectors Using Nelson´s Method. *AIAA Journal,* 14**,** 2355-2356.

19. GALVIS, F. 2002. Estudio del comportamiento dinámico del viaducto Portachuelo basado en medición de vibraciones ambientales. Bogotá.

20. GARCÍA DE JALÓN, J., RODRÍGUEZ, J. & VIDAL, J. 2005. *Aprenda Matlab 7.0 como si estuviera en primero,* Escuela Técnica Superior de Ingenieros Industriales-Universidad Politécnica de Madrid.

21. GHEDAN, H. & NILDEM, T. 2013. Genetic Algorithm Optimization of Space Frame. Tirana, Albania.: Tirana, Albania.

22. GONZÁLEZ, A. G. 2015. *Calibración y validación de modelos de puentes.* UCLV.

23. GONZÁLEZ, D. 2007. *Modelación del puente metálico del ferrocarril km 9.568 del Ramal Santa Clara-Cenizas mediante el Método de los Elementos Finitos.* UCLV.

24. JABER, R. 2007. MATLAB Tutorial. EE313 Signals and Systems.

25. LIU, L. 2004. An Automatic Calibration Strattegy for 3D FE Bridge Models. Cincinnati.

26. LOSA MIRANDA, G. 2015. *Tecnologías de sensorización e identificación modal. Aplicación a la determinación del amortiguamiento estructural.*, UNIVERSIDAD DE VALLADOLID-ESCUELA DE INGENIERÍAS INDUSTRIALES.

27. MAES, K., LOMBAERT, G., VAN NIMMEN, K., PAPADOPOULOS, M. & DE ROECK, G. 2017. Model updating in structural dynamics. *Post-graduate course – Model updating in Structural Dynamics TEAM-VLIR PROJECT "VIBRAS".* Santa Clara, Cuba.

28. MATEMÁTICO, G. 2016. *Matriz jacobiana de una función vectorial* [Online]. Available: (http://www.ub.edu/glossarimateco).

29. MOLINA, S., SALGADO, R., ZAMORA, S. & LAGUNES, E. 2012. Detección de daño en puentes mediante un modelo experimental. XVIII Congreso Nacional de Ingeniería Estructural: Sociedad Mexicana de Ingeniería Estructural.

30. NELSON, R. 1976. Simplified Calculation Of Eigenvector Derivates. *AIAA Journal,* 14**,** 1201-1205.

31. PUENTES, E. C. D. 2005. Investigación del Puente Km. 9,578 de Línea Cenizas. UEB Placetas.

32. RAMÍREZ, R., GÁMEZ, Y., PÉREZ, H. & CHAGOYÉN, E. 2015. Evaluación de un puente de vías férrea mediante ensayos estructurales y modelación computacional. *Obras y Proyectos,* 15**,** 32-43.

33. REYNDERS, E. & DE ROECK, G. System identification and parameter estimation in civil engineering. 2014 Santa Clara, Cuba.

34. REYNDERS, E., DE ROECK, G., GUNDES, P. & SAUVAGE, C. 2007. Damage Identification on the Tilff Bridge by Vibration Monitoring Using Optical Fiber Strain Sensors. *JOURNAL OF ENGINEERING MECHANICS,* Febrero**,** 185-193.

35. RODRÍGUEZ, C., MUÑOZ, D. & NUÑEZ, M. 2007. Optimización aplicada a la calibración y validación de modelos de elementos finitos de puentes. *Ingeniería,* 17**,** 43-59.

36. RODRIGUEZ, M. 2010. *Modelación estructural de un puente metálico ferroviario de viga continua.* UCLV.

37. RODRIGUEZ, M. 2014. *Influencia del comportamiento dinámico en la respuesta de los puentes metálicos de ferrocarril de Cuba para su conservación y/o evaluación.*, UCLV.

38. RODRÍGUEZ, M. 2005. Anáalisis modal operacional: Teoría y Práctica. ESCUELA SUPERIOR DE INGENIEROS.

39. SEQUERA, G. D. & SOLANO, R. L. 2013. Algoritmo para la calibración de modelos estructurales en elementos finitos de puentes usando ANSYS. Bogotá.

40. SEXTOS, A. G. & BALAFAS, G. K. Using the new SAP2000 Open Aplication Programing Interface to develop an interactive front-end for the modal pushover analysis of bridges. 3rd ECCOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering, 2011 Corfu, Greece.

41. SIMOEN, E. 2013. *Uncertainty quantification in finite element model updating.* KU Leuven.

42. SIMOEN, E., DE ROECK, G. & LOMBAERT, G. 2015. Dealing with uncertainty in model updating for damage assessment: A review. *El Sevier,* 56-57**,** 123-149.

43. SORIA, J. M., DÍAZ, I. M., GARCÍA-PALACIOS, J. H. & NORBERTO, I. 2016. Vibration Monitoring of a Steel-Plated Stress-Ribbon Footbridge: Uncertainties in the Modal Estimation. Brasilia.

44. SOVERO, S. K. & MARTEL, C. 2014. *OMA Tests and FEM Updating in Peruvian Archaeological Heritage: Chokepukio y Modal Identification Tests on Archaeological Heritage: The Case of Chokepukio.*, PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ.

45. SUPRIYA, M. & KALYANSHETTI, M. 2013. Dynamic analysis of T-Beam bridge superstructure. *INTERNATIONAL JOURNAL OF CIVIL AND STRUCTURAL ENGINEERING*.

46. TEUGHELS, A. 2003. *Inverse modelling of civil engineerings structures based on operational modal data.* KU Leuven.

ANEXOS

Para el uso de los Anexos es válido aclarar que para su mejor interpretación se debe apoyar en las explicaciones realizadas en los epígrafes correspondientes donde se desarrolla el procedimiento y se explica para qué es el uso de cada código propuesto. Destacar que los códigos descritos a continuación están incompletos, no se les asignaron valores reales a las variables propuestas, pero se aclara en cada caso que es lo que se coloca en cada una. Lo que no tenga especificaciones se mantiene igual.

Matriz Jacobiana en *off*

Anexo 1: Definición del modelo de referencia (data_bridge.m)

```
n_upd_param = n                 % Número de parámetros a actualizar, secciones
x_ref = ;                       % Parámetro de referencia
file = 'C:\Users\...\Mod_Bridge_Ref.sdb';% Dirección archivo *.sdb Referencia
[phi_ref,omega_ref, DOF_ref] = eigfem_bridge(x_ref, file);
modSel    = ;                   % Selección de los modos a analizar
phi_ref   = phi_ref(:,modSel);
omega_ref = omega_ref(modSel);
nDOF   = length(DOF_ref);       % Número de DOF
n_m    = size(modSel,2);        % Total de número de modos a generar
n_node = ;                      % Número de nodos
% Establecer como salvar los datos de referencia
save data_bridge phi_ref omega_ref DOF_ref n_upd_param
```

Anexo 2: Corrida de SAP2000 y obtención de los parámetros modales (eigfem_bridge.m)

```
function [phi,omega,DOF] = eigfem_bridge(x_moduli, file)
if length(x_moduli)==1
    x_moduli=x_moduli*ones(n,1);   % (n) Número de parámetros a actualizar
end
feature('COM_SafeArraySingleDim', 1);
feature('COM_PassSafeArrayByRef', 1);
SapObject = actxserver('sap2000v16.SapObject');
SapObject.ApplicationStart;
SapObject.Hide; % Para que el SAP 2000 no salga por la pantalla
ret = SapObject.SapModel.File.OpenFile(file); % Dirección del modelo SAP2000
SapModel = SapObject.SapModel; % Creando el Objeto SapModel
ret = SapModel.SetModelIsLocked(0); %  Para desbloquear el modelo
%   Switch to units
```

```
%   lb_in_F = 01    lb_ft_F  = 02   kip_in_F = 03   kip_ft_F = 04
%   kN_mm_C = 05    kN_m_C   = 06   kgf_mm_C = 07   kgf_m_C  = 08
%   N_mm_C  = 09    N_m_C    = 10   Ton_mm_C = 11   Ton_m_C  = 12
%   kN_cm_C = 13    kgf_cm_C = 14   N_cm_C   = 15   Ton_cm_C = 16
ret = SapModel.SetPresentUnits(06); %  Seleccionar las unidades de medidas
 %  Creando el modelo de análisis
ret = SapModel.Analyze.CreateAnalysisModel;
Definiendo las propiedades de los materiales creados
MATERIAL_STEEL = 1;
ret = SapModel.PropMaterial.SetMaterial('E1', MATERIAL_STEEL);
       ...
ret = SapModel.PropMaterial.SetMaterial('En', MATERIAL_STEEL);
% Asignando las propiedades mecánicas isotrópicas a los materiales creados
% Las variables x a estimar se corresponden con los módulos de deformación del
   material de los elementos del puente.
   x1=x_moduli(1);...;xn = x_moduli(n); Va desde x1 a xn(secciones a actualizar)
ret = SapModel.PropMaterial.SetMPIsotropic('E1', x1, 0.3, 0.0000117);
 ...
ret = SapModel.PropMaterial.SetMPIsotropic('En', x7, 0.3, 0.0000117);
% Asignando otras propiedades
ret = SapModel.PropMaterial.SetOSteel_1('E1', 22.5553, 37.2653, 32.5, 37.2, 1,
   1, 0.0388, 0.1554, 0.2116, -0.1);
% Asignando propiedades de los materiales peso por unidad de volumen
ret = SapModel.PropMaterial.SetWeightAndMass('E1', 2, 7.847);
 ...
ret = SapModel.PropMaterial.SetWeightAndMass('En', 2, 7.847);
% Sobrescribiendo las propiedades en los elementos seleccionados del modelo
   estructural de la superestructura del puente según los n elementos componentes
Object = 0;
% % Para Tramo I (elemento = número del elemento en SAP2000)
ret = SapModel.FrameObj.SetMaterialOverwrite('elemento', 'E1',Object);
    ...
ret = SapModel.FrameObj.SetMaterialOverwrite('elemento', 'E1',Object);
% % Para Tramo n
ret = SapModel.FrameObj.SetMaterialOverwrite('elemento', 'En',Object);
     ...
ret = SapModel.FrameObj.SetMaterialOverwrite('elemento', 'En',Object);
% Correr Análisis
ret = SapObject.SapModel.Analyze.RunAnalysis();
% Deseleccionando todos los casos y combinaciones de salida
ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput;
% Seleccionando el caso y combinación de salida
ret = SapModel.Results.Setup.SetCaseSelectedForOutput('MODAL');
% Obtención del período modal
```

```
NumberResults = 0;
   Obj = cellstr(' ');
   Elm = cellstr(' ');
   LoadCase = cellstr('MODAL');
   StepType = cellstr('Mode');
   StepNum = zeros(1,1,'double');
   Period = zeros(1,1,'double');
   Frequency = zeros(1,1,'double');
   CircFreq = zeros(1,1,'double');
   EigenValue = zeros(1,1,'double');
[ret, NumberResults, LoadCase, StepType, StepNum, Period, Frequency, CircFreq,
   EigenValue] = ...
SapModel.Results.ModalPeriod(NumberResults, LoadCase, StepType, StepNum, Period,
   Frequency, CircFreq, EigenValue);
n_node = ;                  % Se coloca el número de nodos de la estructura
n_m = length(CircFreq);     % Se coloca el número de modos a analizar
modSel= ;                   % Selección de los modos
omega = CircFreq(modSel).'; % Frecuencias Naturales [rad/s]
% Obtención de las formas modales (coordenadas modales)
   NumberResults = 0;
   Obj = cellstr(' ');
   Elm = cellstr(' ');
   ACase = cellstr('MODAL');
   StepType = cellstr('Mode');
   StepNum = zeros(1,1,'double');
   U1 = zeros(1,1,'double');
   U2 = zeros(1,1,'double');
   U3 = zeros(1,1,'double');
   R1 = zeros(1,1,'double');
   R2 = zeros(1,1,'double');
   R3 = zeros(1,1,'double');
   GroupElm = 2;
[ret, NumberResults, Obj, Elm, ACase, StepType, StepNum, U1, U2, U3, R1, R2, R3]
   = ...
SapModel.Results.ModeShape('ALL', GroupElm, NumberResults, Obj, Elm, ACase,
   StepType, StepNum, U1, U2, U3, R1, R2, R3);
phix = U1;
phiy = U2;
phiz = U3;
phix_=zeros(n_node,n_m);    % intermediate matrix
phiy_=zeros(n_node,n_m);    % intermediate matrix
phiz_=zeros(n_node,n_m);    % intermediate matrix
for ind1=1:n_m
    phix_(:,ind1)= phix(:,1:n_node);
```

```
    phix(:,1:n_node)=[];
    phiy_(:,ind1)= phiy(:,1:n_node);
    phiy(:,1:n_node)=[];
    phiz_(:,ind1)= phiz(:,1:n_node);
    phiz(:,1:n_node)=[];
end
phi=[phix_(:,modSel);phiy_(:,modSel);phiz_(:,modSel)];
DOF=[(1:n_node).'+0.01;...  % 0.01 = x
     (1:n_node).'+0.02;...  % 0.02 = y
     (1:n_node).'+0.03];    % 0.03 = z
ret = SapObject.ApplicationExit(false());    % Cerrando la aplicación Sap2000
      SapModel = 0;
      SapObject = 0;
```

Anexo 3: Inicialización de parámetros (main_bridge.m)

```
    Caso 1: Para el caso de Trust-Region-Reflective Least Squares Algorithm
tic   % Para que cuando termine la corrida obtener el tiempo empleado en segundos
close all; clc;
data_bridge; % Para ejecutar el modelo de referencia
% Restricciones para la actualización de parámetros
E_0   = ;       % Punto de inicio (all stiffness values).Ejemplo = 1.80 * 1e8
E_min = ;       % Lower bound. Ejemplo = 0.20 * E_0
E_max = ;       % Upper bound. Ejemplo = 1.80 * E_0
% MODEL UPDATING
alpha = 1;        % Peso asignado al error en los valores propios
beta  = 1;        % Peso asignado al error en los modos
% Establecer como salvar (archivo .mat)
save upd_param_bridge alpha beta E_0
% Cargar data_bridge (archivo .mat)
load data_bridge phi_ref omega_ref DOF_ref n_upd_param
% Variables que intervienen en la actualización
T_0   = ones(1,n_upd_param);
T_min = E_min./E_0*ones(1,n_upd_param);
T_max = E_max./E_0*ones(1,n_upd_param);
[T_upd] = upd_bridge(T_0,T_min,T_max); % Función para el Model Updating
% RESULTS SUMMARY
load upd_param_bridge E_0
E_0   = T_0.*E_0;
E_upd = T_upd.*E_0;
% Calcular los parámetros modales iniciales
file_upd   =   'C:\Users\...\Upd_Model\Mod_Bridge.sdb';   %Dirección   del   modelo
   actualizado
[phi_0,omega_0,DOF_0]=eigfem_bridge(E_0,file_upd);
```

```
modSel  = ;              % Seleccionar los modos. Ejemplo = modSel=1:6;
phi_0   = phi_0(:,modSel);
omega_0 = omega_0(modSel);
% Calcular los parámetros modales actualizados
[phi_upd,omega_upd,DOF_upd]=eigfem_bridge(E_upd,file_upd);
phi_upd   = phi_upd(:,modSel);
omega_upd = omega_upd(modSel);
% Arreglo de los datos para el ploteo de puntos (iniciales y actualizados)
for i=1:length(phi_ref)
    for k=modSel
    if phi_0(i,k)<=0 && phi_ref(i,k)<=0 || phi_0(i,k)>=0 && phi_ref(i,k)>=0
        phi_0(i,k)=phi_0(i,k);
    else
        phi_0(i,k)=-phi_0(i,k);
    end
    end
end
for i=1:length(phi_ref)
    for k=modSel
    if phi_upd(i,k)<=0 && phi_ref(i,k)<=0 || phi_upd(i,k)>=0 && phi_ref(i,k)>=0
        phi_upd(i,k)=phi_upd(i,k);
    else
        phi_upd(i,k)=-phi_upd(i,k);
    end
    end
end
% Posición de los sensores y dirección de análisis. Ejemplo:
DOF_sensors = [horzcat(1,…,30).'+0.01;... % DOFs where sensors are located
               horzcat(1,…,30).'+0.02;...
               horzcat(1,…,30).'+0.03;...
               horzcat(1,…,30).'+0.04;...
               horzcat(1,…,30).'+0.05;...
               horzcat(1,…,30).'+0.06];
% Calcular la matriz L
L_ref = selectdof(DOF_ref,DOF_sensors);
L_0   = selectdof(DOF_0,DOF_sensors);
% Presentación de los resultados de la actualización
disp('Elasticity modulus initial [kN/m2]')
fprintf('%15.2f',E_0)
fprintf('\n')
disp('Elasticity modulus updated [kN/m2]')
fprintf('%15.2f',E_upd)
fprintf('\n')
```

```
fprintf('\n')
disp('Identified natural frequencies [Hz]')
fprintf('%9.2f',omega_ref/2/pi)
fprintf('\n')
disp('Calculated natural frequencies - (initial) [Hz]')
fprintf('%9.2f',omega_0/2/pi)
fprintf('\n')
disp('Calculated natural frequencies - (updated) [Hz]')
fprintf('%9.2f',omega_upd/2/pi)
fprintf('\n')
disp('Error natural frequencies - (initial) [Hz]')
fprintf('%9.2f',(omega_0-omega_ref)/2/pi)
fprintf('\n')
disp('Error natural frequencies - (initial) [%]')
fprintf('%9.2f',(omega_0-omega_ref)./omega_ref*100)
fprintf('\n')
disp('Error natural frequencies - (updated) [Hz]')
fprintf('%9.2f',(omega_upd-omega_ref)/2/pi)
fprintf('\n')
disp('Error natural frequencies - (updated) [%]')
fprintf('%9.2f',(omega_upd-omega_ref)./omega_ref*100)
fprintf('\n')
fprintf('\n')
disp('Stiffness initial [kN/cm^2]')
fprintf('%15.2f',E_0)
fprintf('\n')
disp('Stiffness updated [kN/cm^2]')
fprintf('%15.2f',E_upd)
fprintf('\n')
fprintf('\n')
disp('MAC (initial %) [-]')
fprintf('%15.5f',diag(mac(L_0*phi_0,L_ref*phi_ref))*100)
fprintf('\n')
disp('MAC (updated %) [-]')
fprintf('%15.5f',diag(mac(L_0*phi_upd,L_ref*phi_ref))*100)
fprintf('\n')
% Cargar results_bridge
load results_bridge T_iter resnorm_iter
% Ploteo de la norma de los residuos para la función objetivo
figure
plot(resnorm_iter)
xlabel('Iteration number[-]')
ylabel('objetive funtion[-]')
```

```
title('Objective function residuals')
set(gca,'fontsize',11)
% Ploteo de las frecuencias naturales de referencia, iniciales y actualizadas
figure
plot(modSel,omega_ref/2/pi,'r--o',modSel,omega_0/2/pi,'k--.',modSel,...
omega_upd/2/pi,'b:*')
hold on
title('Natural Frequencies vs Number of Modes')
xlabel('Number of Mode')
ylabel('Natural Frequency (Hz)')
legend('reference','initial','updated','Location','southeast')
set(gca,'fontsize',11)
% Definir los nodos y la posición para graficar.
x_1=[coordenadas de los nodos analizados en x];
y_1=[coordenadas de los nodos analizados en y];
column_nodes_1 =  nodos analizados;
%% Ploteo de los modos para las columnas de nodos definidas
for i=modSel
  figure
    xlim([-inf,inf])
    ylim([-inf,inf])
    switch i
        case 1
            plot(x_1,phi_ref(30.+column_nodes_1,i),'r-o')
            hold on
            plot(x_1,phi_0(30.+column_nodes_1,i),'b:o')
            plot(x_1,phi_upd(30.+column_nodes_1,i),'k--*')
            hold off
            ylabel('Displacement "y"');
        case 2
            plot(x_1,phi_ref(30.+column_nodes_1,i),'r-o')
            hold on
            plot(x_1,phi_0(30.+column_nodes_1,i),'b:o')
            plot(x_1,phi_upd(30.+column_nodes_1,i),'k--*')
            hold off
            ylabel('Displacement "y"');
        case 3
            plot(x_1,phi_ref(30.+column_nodes_1,i),'r-o')
            hold on
            plot(x_1,phi_0(30.+column_nodes_1,i),'b:o')
            plot(x_1,phi_upd(30.+column_nodes_1,i),'k--*')
            hold off
            ylabel('Displacement "y"');
```

```
        case 4
            plot(x_1,phi_ref(60.+column_nodes_1,i),'r-o')
            hold on
            plot(x_1,phi_0(60.+column_nodes_1,i),'b:o')
            plot(x_1,phi_upd(60.+column_nodes_1,i),'k--*')
            hold off
            ylabel('Displacement "z"')
        case 5
            plot(x_1,phi_ref(30.+column_nodes_1,i),'r-o')
            hold on
            plot(x_1,phi_0(30.+column_nodes_1,i),'b:o')
            plot(x_1,phi_upd(30.+column_nodes_1,i),'k--*')
            hold off
            ylabel('Displacement "y"')
            otherwise
            plot(x_1,phi_ref(60.+column_nodes_1,i),'r-o')
            hold on
            plot(x_1,phi_0(60.+column_nodes_1,i),'b:o')
            plot(x_1,phi_upd(60.+column_nodes_1,i),'k--*')
            hold off
            plot(x_1,phi_ref(60.+column_nodes_2,i),'r-o')
            hold on
            plot(x_1,phi_0(60.+column_nodes_2,i),'b:o')
            plot(x_1,phi_upd(60.+column_nodes_2,i),'k--*')
            hold off
            ylabel('Displacement Torsional "z"')
    end; %switch
    xlabel('Distance')
    title(['Mode ' int2str(i)])
    legend('reference','initial','updated','Location','southeast')
end
savefig((1:((max(modSel))+2)),'All updating figures')
toc % Para que cuando termine la corrida obtener el tiempo empleado
```

Caso 2: Para el caso de Levenberg-Marquardt Method

Solo cambia lo siguiente:

```
% PARAMETER VALUES UPDATING
E_0 = ;           % Punto de inicio (all stiffness values).Ejemplo = 1.80 * 1e8
MODEL UPDATING
alpha = 1;        % Peso asignado al error en los valores propios
beta  = 1;        % Peso asignado al error en los modos
save upd_param_bridge alpha beta E_0
% Cargar data_bridge (archivo .mat)
```

```
load data_bridge phi_ref omega_ref DOF_ref n_upd_param
T_0 = ones(1,n_upd_param);
[T_upd] = upd_bridge(T_0); ;        % Función para el Model Updating
```

Anexo 4: Actualización del modelo (upd_bridge.m)

```
    Caso 1: Para el caso de Trust-Region-Reflective Least Squares Algorithm
function [T_upd]=upd_bridge(T_0,T_min,T_max)
TolFun = ;     % Tolerancia en la función objetivo. Ejemplo = 1e-12
TolX   = ;     % Tolerancia en los parámetros. Ejemplo = 1e-8
% Especificación de las opciones
options=optimset('Display','iter','TolFun',TolFun,'TolX',TolX,...
    'Jacobian','off','OutputFcn',@outfun);
% Optimización no lineal
[T_upd,resnorm,residual,exitflag,output]=...
    lsqnonlin(@objfun,T_0,T_min,T_max,options);
function [res]=objfun(T) % Función objetivo
              Caso 2: Para el caso de Levenberg-Marquardt Method
function [T_upd]=upd_bridge(T_0)
TolFun    = ;     % Tolerancia en la función objetivo. Ejemplo = 1e-12
TolX      = ;     % Tolerancia en los parámetros. Ejemplo = 1e-8
initdamp = ;     % Initial damping of search direction lambda parameter
% Especificación de las opciones
options=optimoptions(@lsqnonlin,'Algorithm','levenberg-
   marquardt','ScaleProblem','Jacobian','InitDamping',initdamp,'Display','iter
   -
   detailed',...'Jacobian','off','TolFun',TolFun,'TolX',TolX,'OutputFcn',@outf
   un);
load data_bridge n_upd_param
% Optimización no lineal
[T_upd]=...  lsqnonlin(@objfun,T_0,-inf*ones(1,n_upd_param)...
    ,+inf*ones(1,n_upd_param),options);
function [res]=objfun(T)
         -----A partir de aquí se mantiene igual para ambos casos-----
% Establecer como salvar
load upd_param_bridge_mdof E_0
Evec = T * E_0; % Variable a utilizar para la actualización
% Cargar los datos experimentales o de referencia
load data_bridge phi_ref omega_ref DOF_ref
% Obtención de los parámetros modales utilizando un modelo
file   =   'C:\Users\...\Upd_Model\Mod_Bridge.sdb';       %Dirección   del   modelo
   actualizado
% Calculate modal parameters using model
model = 3; % 1 = analytical, 2 = ANSYS, 3 = OAPI SAP2000 MatLab
switch model
```

```
    case 1 % ANALYTICAL MODEL
        mvec=m_ref*ones(nfloor,1);
        [phi,omega]=eigfem_mdof(mvec,kvec);
    case 2 %  ANSYS MODEL
        ansys_dir='..\..\ansys\exercise2';
        [phi,omega]=getansys(kvec,ansys_dir);
    case 3 % OAPI SAP2000 MatLab MODEL
        [phi,omega,DOF] = eigfem_bridge(Evec,file);
end
% Posición de los sensores y dirección de análisis. Ejemplo:
DOF_sensors = [horzcat(1,…,30).'+0.01;... % DOFs where sensors are located
               horzcat(1,…,30).'+0.02;...
               horzcat(1,…,30).'+0.03;...
               horzcat(1,…,30).'+0.04;...
               horzcat(1,…,30).'+0.05;...
               horzcat(1,…,30).'+0.06];
% Matriz L
L_ref = selectdof(DOF_ref,DOF_sensors);
L     = selectdof(DOF,DOF_sensors);
% Mode matching/Trabajo con los modos
MAC = mac(L_ref*phi_ref,L*phi); % Calcular la matriz MAC
treshold = 0.5;                 % Treshold for mode matching (mayor o igual a)
mode_ref_sel=(1:size(MAC,1))';  % No. De modos medidos
mode_sel=zeros(size(MAC,1),1);  % No. Modos calculados
for ind1=1:length(mode_ref_sel)
   [max_,ind_max]=max(MAC(ind1,:));
    if max_>=treshold
        mode_sel(ind1)=ind_max; % Match found
    else
        disp(['No mode match found for measured mode ' int2str(ind1) '.'])
    end
end
mode_ref_sel(mode_sel==0,:)=[];
mode_sel(mode_sel==0,:)=[];
if length(unique(mode_sel))~=length(mode_sel)
    disp('At least one calculated mode is matched to multiple measured modes.')
end
phi_ref=phi_ref(:,mode_ref_sel);omega_ref=omega_ref(mode_ref_sel);
phi=phi(:,mode_sel);omega=omega(mode_sel);
nm=length(omega);
% Calcular los factores de escalas de las formas modales
gamma=zeros(nm,1);
for ind1=1:nm
```

```
gamma(ind1)=((L_ref*phi_ref(:,ind1)).'*L*phi(:,ind1))/norm(L_ref*phi_ref(:,ind1
    ),2)^2;
end
% Calcular los valores propios
lambda = omega;
lambda_ref = omega_ref;
% Calcular los residuos
load upd_param_brid_mdof alpha beta
res=(sqrt(alpha/2).*((lambda-lambda_ref)./lambda_ref).^2);
for ind1=1:nm
    wf=sqrt(beta/2)/(norm(gamma(ind1)*L_ref*phi_ref(:,ind1),2));
 res=[res;(wf.*(L*phi(:,ind1)(gamma(ind1)*L_ref*phi_ref(:,ind1)))).^2];
end
% Función de salida
function stop=outfun(T,optimValues,state)
stop=false;
% Salvar los resultados de las iteraciones
if strcmp(state,'init')
    T_iter=T';
    resnorm_iter=optimValues.resnorm;
    save results_bridge T_iter resnorm_iter
elseif strcmp(state,'iter')
    load results_bridge T_iter resnorm_iter
    T_iter=[T_iter;T'];
    resnorm_iter=[resnorm_iter;optimValues.resnorm];
    save results_bridge T_iter resnorm_iter
end
```

Anexo 5: Análisis de sensibilidad (sensitivity_analysis.m)

```
% Modelo de referencia
E_ref = ; % Establecer el valor inicial para el daño inducido. Ejemplo=2.0 * 1e8
E_0 = ;   % Para la estructura no dañada. Ejemplo 2.1 * 1e8
% Calcular los parámetros modales
load data_bridge.mat
file_upd = 'C:\Users...\Upd_Model\Mod_Bridge.sdb';   % Escribir la dirección
    exacta del archivo *.sdb
[phi_0,omega_0,DOF]=eigfem_bridge(E_0,file_upd);
modSel = ;       % Selección delos modos a analizar. Ejemplo = modSel=1:6;
phi_0=phi_0(:,modSel);
omega_0=omega_0(modSel);
% Posición de los sensores y dirección de análisis. Ejemplo:
DOF_sensors = [horzcat(1,…,30).'+0.01;... % DOFs where sensors are located
               horzcat(1,…,30).'+0.02;...
```

```
                horzcat(1,…,30).'+0.03;...
                horzcat(1,…,30).'+0.04;...
                horzcat(1,…,30).'+0.05;...
                horzcat(1,…,30).'+0.06];
% Matriz L
L_ref = selectdof(DOF_ref,DOF_sensors);
L_0   = selectdof(DOF,DOF_sensors);
% Calcular AUTO-MAC
autoMAC=mac(L_0*phi_ref(:,modSel),L_0*phi_ref(:,modSel));
% Establecer las propiedades de salida de la figura con la matriz MAC
thisFig=figure;
gcfPos=get(gcf,'position');
bar3color(autoMAC)
view([0 90])
axis equal
hold on
xlim([0.4 size(autoMAC,1)+0.6])
ylim([0.4 size(autoMAC,1)+0.6])
set(gca,'clim',[0 1])
colorbar;
box on
colormap((hot))
labels=[];
for ind1=1:length(modeSel)
    labels=strvcat(labels,int2str(modSel(ind1)))
end
set(gca,'Xtick',1:length(modSel),'XTickLabel',labels)
set(gca,'Ytick',1:length(modSel),'YTickLabel',labels)
```

Matriz Jacobiana en *on*

Anexo 6: Definición del modelo de referencia (data_bridge.m)

```
n_upd_param = n;                % Número de parámetros a actualizar, secciones
x_ref = ones(n_upd_param,1)*2000*1e05;    % Parámetro de referencia
% Correr SAP2K con un Módulo de Young inicial( caso DEAD para generar K y M)
file = 'C:\Users\...\Ref_Model\Mod_Bridge_Ref.sdb'; % Dirección del modelo de
   referencia
run_SAP2k_on(x_ref,file)              % Ejecutar el modelo de referencia
% Ensamblar la matriz K y M generadas por SAP2000
filex = 'C:\Users\...\Ref_Model\Mod_Bridge_Ref.TXA'; % Dirección del fichero
[K_ref,M_ref,~,DOF_ref]=asmkm_sap2k(filex);
nDOF   = length(DOF_ref);          % Número de DOF
```

```
modSel = ;
n_m    = size(modSel,2);          % Total de modos a generar
n_node = ;                        % Número de nodos
% Obtención de los parámetros a través de STABIL 2.0
[phi_ref,omega_ref] = eigfem(K_ref,M_ref,n_m);
phi_ref   = phi_ref(:,modSel);
omega_ref = omega_ref(modSel);
% Establecer como salvar los parámetros de referencia
save data_bridge n_upd_param phi_ref omega_ref DOF_ref
```

Anexo 7: Corrida de SAP2000 (run_SAP2k.m)

```
function [] = run_SAP2k_on(x_moduli, file)
if length(x_moduli)==1
    x_moduli=x_moduli*ones(n,1);
end
   ----------(Se mantiene lo mismo que para eigfem_bridge Anexo 2)----------
% Run Analysis (Correr Análisis)
ret = SapModel.Analyze.SetRunCaseFlag('DEAD', 1);
ret = SapModel.Analyze.SetSolverOption_1(1,0,false(),'DEAD');  % Salida de las
   matrices M y K
ret = SapObject.SapModel.Analyze.RunAnalysis();
% Deseleccionando todos los casos y combinaciones de salida
ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput;
```

Anexo 8: Inicialización de parámetros (main_bridge.m)

```
% Calcular los parámetros modales iniciales
file  = 'C:\Users\...\Upd_Model\Mod_Bridge.sdb';  % Dirección del archivo .sdb
filex = 'C:\Users\...\Upd_Model\Mod_Bridge.TXA';  % Dirección del fichero
run_SAP2k(E_0,file);                  % Correr SAP2000 para obtener K y M
[K_0,M_0,~,DOF_0]=asmkm_sap2k(filex); % Ensamblar K y M
[phi_0,omega_0] = eigfem(K_0,M_0);    % Calcular los parámetros modales
modSel  =;                            % Número de modos a trabajar
phi_0   = phi_0(:,modSel);
omega_0 = omega_0(modSel);
% Calcular los parámetros modales actualizados
run_SAP2k(E_upd, file);                     % Correr SAP2000 para obtener K y M
[K_upd,M_upd,~,DOF_upd]=asmkm_sap2k(filex); % Ensamblar K y M
[phi_upd,omega_upd] = eigfem(K_upd,M_upd);  % Calcular los parámetros modales
phi_upd   = phi_upd(:,modSel);
omega_upd = omega_upd(modSel);
```

Anexo 9: Actualización del modelo (upd_bridge.m)

```
                -----Calculate Jacobian (Nelson's Method)-----
npar = length(T);              % Número de parámetros a actualizar
nd = length(DOF_ref);          % Número de sensores
ndof = size(phi,1);            % Número de DOF
jac_lambda = zeros(nm,npar);
jac_vec = zeros(size(DOF_sensors,1)*nm,npar);
for ind1=1:npar
% Calcular las derivadas de la matriz rigidez para los parámetros actualizados
      T0 = T(ind1);
      delta_T = 1e-8 * T0;   % Step applied for finite difference calculation
      xl = E_0*ones(npar,1); xl(ind1) = xl(ind1)*(T0-delta_T/2);   % lower value
   for Young's module
      xu = E_0*ones(npar,1); xu(ind1) = xu(ind1)*(T0+delta_T/2);   % upper value
   for Young's module
 % Correr modelo para xl
 % Run SAP2K model for lower value for Young's module
      run_SAP2k(xl,file);
 % Leer las matrices de masa y rigidez para el modelo inicial con xl
      [Kl,Ml,~,~]=asmkm_sap2k(filex); %nDOF=length(DOFl);
 % Correr modelo para xu
 % Run SAP2K model for upper value for Young's module
      run_SAP2k(xu,file);
 % Leer las matrices de masa y rigidez para el modelo inicial con xu
      [Ku,Mu,~,~]=asmkm_sap2k(filex);
 % Derivadas dK/dx y dM/dx
      dK = (Kl-Ku) / delta_T;
      dM = (Ml-Mu) / delta_T;              % En este caso es =0
      clear delta_T Kl Ku Ml Mu DOFl xl xu;
    for ind2=1:nm
 % Jacobiano (valores propios)
        jac_lambda(ind2,ind1)=...
            sqrt(alpha/2)/lambda_ref(ind2)*phi(:,ind2).'*dK*phi(:,ind2);
        % Vectores propios del Jacobiano
        % Original sistema de matices
        A=K-lambda(ind2)*M;
        b=-(dK-phi(:,ind2).'*(dK-lambda(ind2)*dM)*phi(:,ind2)*M)*phi(:,ind2);
        % Modificando el sistema de matrices
        [~,ksel]=max(abs(phi(:,ind2)));
        A(ksel,:)=0; A(:,ksel)=0; A(ksel,ksel)=1;
        b(ksel)=0;
        wf=sqrt(beta/2)/norm(gamma(ind2)*phi_ref(:,ind2),2);  % weighting factor
```

```
        jac_vec((ind2-1)*size(DOF_sensors,1)+(1:size(DOF_sensors,1)),ind1)=...
            wf*L*(eye(ndof)-phi(:,ind2)*phi(:,ind2).'*M)*(A\b);
    end
end
jac=[jac_lambda;jac_vec];
```

Anexo 10: Obtención de matrices K, M, T (asmkm_sap2k.m)

```
function [K,M,T,DOF]=asmkm_sap2k(file)
% Exportar las matrices de masa M y rigidez K, Vector DOF y constraint desde
   SAP2000
% Cargar archivo
file = 'C:\Users\...\Modelo_Ref\Mod_Bridge.TXA';  % Este descriptor de fichero
   puede cambiarse
[pathstr,name,ext] = fileparts(file);
% Leer. TXK. lee el archivo *.TXK Matriz K generada por SAP2000
filename=fullfile(pathstr,[name '.TXK']);
fileID=fopen(filename);
data=textscan(fileID,'%f%f%f','Delimiter','\t','headerlines',1);
fclose(fileID);
K0 = [data{1},data{2},data{3}];    % se guarda como la matriz rigidez inicial
clear filename fileID data
% Leer. TXM. Lee el archivo *.TXM matriz M generada por SAP2000
filename=fullfile(pathstr,[name '.TXM']);
fileID=fopen(filename);
data=textscan(fileID,'%f%f%f','Delimiter','\t','headerlines',1);
fclose(fileID);
M0 = [data{1},data{2},data{3}];
clear filename fileID data
% Leer. TXE. Lee el archivo *.TXE Número de ecuaciones generadas por SAP2000
filename=fullfile(pathstr,[name '.TXE']);
fileID = fopen(filename);
data=textscan(fileID,'%s%s%s%s%s%s','Delimiter','\t','headerlines',1);
fclose(fileID);
DOF0=[str2double(data{1})        str2double(data{2})        str2double(data{3})
   str2double(data{4}) str2double(data{5}) str2double(data{6})];
DOF0(1:2:end,7) = DOF0(2:2:end,2);
DOF0(2:2:end,:) = [];
DOF0(isnan(DOF0(:,1)),1)=max(DOF0(:,1))+(1:sum(isnan(DOF0(:,1)))).';
clear filename fileID data data_dummy
% Leer. TXC. lee el archivo *.TXC Constrains generados por SAP2000
try
    flag=0;
    filename=fullfile(pathstr,[name '.TXC']);
    fileID = fopen(filename);
```

```
    data=textscan(fileID,'%f%f%f','Delimiter','\t','headerlines',1);
    fclose(fileID);
    C0 = [data{1},data{2},data{3}];
    clear filename fileID data
catch
    flag=1;
    clear filename fileID
end
DOF
dof=reshape(DOF0(:,2:end).',[6*size(DOF0,1),1]);        % DOF vector of SAP
DOF=reshape((repmat(DOF0(:,1),[1,6])+repmat([0.01    0.02    0.03    0.04    0.05
   0.06],[size(DOF0,1),1])).',[6*size(DOF0,1),1]); % DOF vector of STABIL
Masterdof=dof(dof>0);
MasterDOF=DOF(dof>0);
[Masterdof,I]=sort(Masterdof);
[Masterdof,J]=unique(Masterdof,'last');
MasterDOF=MasterDOF(I);
MasterDOF=MasterDOF(J);
[~,I]=unselectdof(DOF,MasterDOF);
SlaveDOF=DOF(I);
Slavedof=dof(I);
I=Slavedof==0;
Slavedof(I)=[];
SlaveDOF(I)=[];
dof=[Masterdof; Slavedof];
DOF=[MasterDOF; SlaveDOF];
K Matrix
I=match(K0(:,1),Masterdof);
J=match(K0(:,2),Masterdof);
K=sparse(I,J,K0(:,3));
K=K+K.'-spdiags(diag(K),0,sparse(length(Masterdof),length(Masterdof)));
M Matrix
I=match(M0(:,1),Masterdof);
J=match(M0(:,2),Masterdof);
M = sparse(I,J,M0(:,3));
M = M + M.'-spdiags(diag(M),0,sparse(length(Masterdof),length(Masterdof)));
T matrix
if flag==0
    I1=match(C0(:,1),Slavedof);
    I2=match(C0(:,2),Masterdof);
    C=[SlaveDOF(I1) MasterDOF(I2) C0(:,3)];
    J=match(Slavedof,Masterdof);
    Constr=zeros(length(SlaveDOF),15);
```

```
    nConstr=size(Constr,1);
    Constr(:,2)=-1;
    Constr(:,3)=SlaveDOF;
    Constr(:,5:2:end)=MasterDOF(1);
    for i=1:nConstr
        if J(i)~=0
            Constr(i,4)=1;
            Constr(i,5)=MasterDOF(J(i));
        else
            ind=C(:,1)==SlaveDOF(i);
            num=sum(ind);
            Constr(i,4:2:4+2*(num-1))=C(ind,3).';
            Constr(i,5:2:5+2*(num-1))=C(ind,2).';
        end % if J(i)
    end  % for i = 1:nConstr
    T = tconstr(Constr,DOF);
else
    T = zeros(size(K));
end % if flag==0
% save FE matrices
clear DOF0 I J K0 M0 MasterDOF Masterdof SlaveDOF Slavedof dof ext file flag
   name pathstr
% load asmkm_sap2000.mat;
```