UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

CÁLCULO NUMÉRICO DE LOS MODOS DE VIBRACIÓN EN UN PANEL COMPUESTO TIPO PANAL DE ABEJA MEDIANTE ANÁLISIS ISOGEOMÉTRICO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL MECÁNICA

CAMILA FRANCISCA VELÁSQUEZ SALINAS

PROFESOR GUÍA:

ALEJANDRO ORTIZ BERNARDIN

MIEMBROS DE LA COMISIÓN

VIVIANA MERUANE NARANJO

WILLIAMS CALDERÓN MUÑÓZ

SANTIAGO DE CHILE

2015

CÁLCULO NUMÉRICO DE LOS MODOS DE VIBRACIÓN EN UN PANEL COMPUESTO TIPO PANAL DE ABEJA MEDIANTE ANÁLISIS ISOGEOMÉTRICO

La constante búsqueda de la optimización de recursos en diversas áreas de la ingeniería a nivel mundial crea la necesidad del desarrollo de nuevos materiales y configuraciones estructurales. Es en este contexto, que nace la gama de estructuras compuestas, que corresponden a unidades constituidas por dos capas delgadas, envolviendo al centro de un material más liviano. Como resultado, se obtiene una estructura liviana pero con alta rigidez a la flexión, que tiene aplicaciones en diversas áreas de la ingeniería (aeroespacial, energética, transporte, entre otros). Por esto, resulta primordial contar con metodologías robustas para determinar y controlar la vida útil de los mismos.

Este documento postula una herramienta para perfeccionar el trabajo realizado por un grupo de ingenieros del Departamento de Ingeniería Mecánica de la Universidad de Chile, en el desarrollo de un método de detección de delaminación en paneles compuestos tipo panal de abeja. En particular, se explora el uso análisis isogeométrico (IGA) con funciones de forma tipo NURBS (Non Uniform Rational B-Splines), las cuales poseen suavidad superior a las del método de elemento finito tradicional (FEM). Esta mayor suavidad debiera resultar en un mejor cálculo numérico de los modos de vibración del panel compuesto a estudiar.

Dicho lo anterior, el objetivo general es calcular numéricamente los modos de vibración de una placa compuesta tipo panal de abeja, mediante IGA, con funciones de forma tipo NURBS. Para lograrlo, se plantean diversos objetivos específicos, entre los cuales se encuentra el entrenamiento en la utilización del método, cálculo de modos de vibración del panel compuesto y cálculo de correlación numérico experimental de modos obtenidos.

Se considera una metodología iterativa, donde se seleccionan y varían los parámetros que definen el modelo (orden de las NURBS y tamaño de malla) en búsqueda del óptimo, que se define en base a precisión.

Finalmente, se determina que el mejor modelo contiene NURBS de orden 12 en una malla de 50x30 elementos, con el cual se obtienen correlaciones casi perfectas en todos los modos a excepción del modo N°5, para el cual solo se alcanza un 93,9 % y no directamente sino que se requiere la aplicación de un procedimiento posterior de manipulación de datos.

Además, se tiene un mayor requerimiento computacional, lo que pudiere mejorarse con algoritmos más eficientes de integración o la optimización del programa. Finalmente, pese a que no se constituye como un mejor método que FEM para análisis modal, IGA sí tiene un mejor desempeño en otro tipo de problemas (ej. pandeo de Euler).

A mis abuelos Óscar y Mirto

Agradecimientos

Primero que todo, debo agradecer a la mujer más grande: mi madre, Isabel. Por estar ahí siempre, por enseñarme tantas cosas, por ser como eres. Sobre todas las cosas, por haberme convertido en la mujer que soy.

Gracias también a mi hermano, Felipe, por la paciencia y apoyo en todo momento, por ser fiel compañero y el mejor amigo. También a mi hermana, Andrea, por estar siempre ahí para el minuto de distención, recibirme siempre en tu casa y en tus panoramas; pero sobretodo, por traer a mi vida a Julieta y Olivia.

Gracias a la familia: tíos, primos, todos. También la familia que no es biológica pero es del alma, gracias mi segunda madre, Marilú, por recibirme cada vez que lo necesité. Gracias a mis hermanas del corazón, Cassandra y Claudia, por estar siempre ahí para mí. También le agradezco a mi papá por los buenos años y las buenas enseñanzas.

¿Y cómo no mencionar a los amigos? A esos que han sufrido conmigo las peripecias universitarias, a aquellos que están ahí con una cerveza fría para celebrar cada viernes que hemos pasado otra semana. Gracias a todos: Valbort, Catalina, Octavio, Sergio, Claudio, Javier y Álvaro. También a los amigos de la oficina, gracias a Daniela, porque me puso en sus agradecimientos.

No puedo no mencionar a mi abuelo Óscar, sin él, nada de esto hubiese sido posible. Gracias por convencerme de que vale la pena luchar por los sueños y por guardarme cada ápice de la PSU que se asomaba en algún diario o revista. Gracias a mi abuela Mirto, por las llamadas semanales, por el consejo sabio, por rezarle a San Expedito para que pasara mecánica (¡y lo pasé!), por tantas cosas. Espero que donde quiera que estén ahora, puedan ver que lo logré, tendrán su nieta ingeniera.

Gracias a Les Piegl y Wayne Tiller, porque sin The NURBS Book aun no entendería las Rational B-Splines.

Por último, pero no menos importante, gracias a Víctor, por estar junto a mí a lo largo de este proceso que ha sido todo, menos fácil. Por convencerme de que podía, por amarme.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1.- INTRODUCCIÓN

La constante búsqueda de la optimización de recursos en diversas áreas de la ingeniería a nivel mundial crea la necesidad del desarrollo de nuevos materiales configuraciones estructurales. Es en este contexto, que nace la gama de estructuras compuestas, que corresponden a unidades constituidas por dos capas delgadas, envolviendo al centro compuesto de un material más liviano. Como resultado, se obtiene una estructura liviana pero con alta rigidez a la flexión.

Dadas las características de los paneles compuestos, tienen diversas aplicaciones que van desde satélites, aviones, barcos, automóviles, vagones de ferrocarril, sistemas de energía eólica hasta la construcción de puentes, entre otras [1]. Dicho esto, resulta crucial contar con metodologías robustas para determinar y controlar la vida útil de los mismos.

Uno de los problemas más relevantes en estos paneles compuestos corresponde a la separación o despegue de las placas con el centro, lo que resulta altamente perjudicial en su comportamiento mecánico. Para localizar la presencia de estos puntos de separación, se pueden utilizar técnicas de monitoreo de fallas, las cuales requieren calcular los modos de vibración en el panel con mucha exactitud.

En particular, este documento presenta una herramienta para perfeccionar el trabajo realizado por un grupo de ingenieros del Departamento de Ingeniería Mecánica de la Universidad de Chile, en el desarrollo de un método de detección de separación de placas en paneles compuestos tipo panal de abeja [2].

Para el documento mencionado, se utiliza el método de elemento finito tradicional para el cálculo numérico de los modos de vibración y se comparan dichos modos con mediciones experimentales. Debido a las características geométricas de los paneles compuestos tipo panal de abeja, el uso del método de elemento finito tradicional no logra capturar con mucha exactitud algunos modos de vibración. En particular, en esta memoria se explorará el uso del método IGA (IsoGeometric Analysis), que utiliza funciones de forma tipo NURBS (Non Uniform Rational B-Splines), las cuales poseen suavidad muy superior a las funciones de forma del método de elemento finito tradicional. Esta mayor suavidad debería resultar en una mejoría en el cálculo numérico de los modos de vibración del panel compuesto a estudiar.

Para dar continuidad al estudio, se utiliza el mismo modelo para el cual se han calculado los modos con método de elemento finito tradicional, el cual se especificará en la sección correspondiente en el desarrollo de esta memoria. Además, se cuenta con los datos experimentales, para comparar, entregados por los autores del mencionado estudio.

En función de lo dicho, se determinan los objetivos generales y específicos de esta memoria, lo cuales se muestran a continuación.

Objetivo General

Calcular numéricamente los modos de vibración de una placa compuesta tipo panal de abeja mediante método de análisis isogeométrico (IGA), con funciones de forma tipo NURBS (Non Uniform Rational B-Splines).

Objetivos específicos

- Resolver modelos numéricos simples mediante análisis isogeométrico.
- Calcular modos de vibración de una placa simple mediante análisis isogeométrico y validar el modelo mediante comparación con modos calculados en Mechanical APDL de ANSYS®.
- Programar un modelo paramétrico de la placa compuesta tipo panal de abeja con funciones NURBS para análisis isogeométrico (IGA). Luego calcular modos de vibración y frecuencias naturales, utilizando el software Matlab®.
- Comparar los modos de vibración numéricos con modos medidos de manera experimental, mediante correlación.

Los alcances de este documento abarcan solo hasta el cálculo numérico de los modos de vibración para la placa mencionada y no se considera la aplicación de los mismos en ningún programa de monitoreo de fallas.

2.- ANTECEDENTES

2.1.- Placas compuestas tipo panal de abeja

Las placas compuestas tipo panal de abeja son estructuras formadas por dos placas planas delgadas, que envuelven una estructura más liviana en formación de panal de abeja hexagonal, como puede verse en la Figura 2.1.

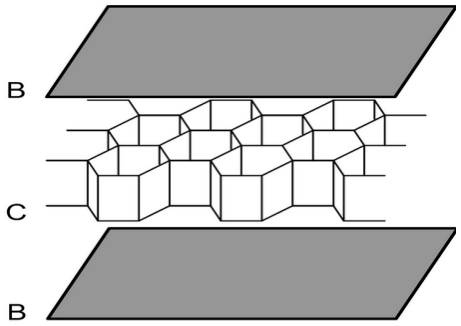


Figura N° 2.1: Modelo de placa compuesta tipo panal de abeja.

La unión entre las placas B y el centro (C) se realiza con diversos tipos de capas adhesivas, en particular.

Los usos y aplicaciones de estas estructuras varían de acuerdo a los materiales utilizados, mas en general se presentan como una alternativa a perfiles de alta rigidez a la flexión en aplicaciones que requieren reducir el peso de la estructura total.

2.2.- Funciones NURBS

Las funciones de forma tipo NURBS (Non Uniform Rational B-Splines) tienen gran importancia pues son ampliamente utilizadas por programas CAD (Computer Asisted Design) que son, a su vez, cada vez más requeridos en el área de la ingeniería.

Para comprender la definición de las NURBS, se debe comprender su base. Las B-Splines son curvas polinomiales no interpoladas que se definen por tramos. Se crean en base a una serie de puntos de control ($P_i, i = 1 \dots n$), el grado polinomial (p) y un vector de puntos ($\Xi = \xi_1, \dots, \xi_{n+p+1}$). Este vector de puntos corresponde una serie de coordenadas paramétricas, que dividen la B-Spline en segmentos, en el caso de que los puntos sean equiespaciados, el vector de puntos se dice uniforme.

Una B-Spline se define con funciones de base. Para definir una B-Spline, se utiliza la formula recursiva de Cox-deBoor, que comienza para p=0 y dice:

$$N_{i,0} = \begin{cases} 0 \ \xi_i \leq \xi \leq \xi_{i+1} \\ 1 \ \ otro\ caso \end{cases} \tag{2.1}$$

Para $p \geq 1$:

$$N_{i,p} = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{2.2}$$

donde :

ξ: Punto de evaluación.

ξ_i: i-ésimo punto del vector de puntos.

p: grado del polinomio.

$N_{i,p-1}$: i-ésima función de forma de grado p-1.

Nótese que las funciones de base son de tipo C^{∞} entre dos puntos y C^{p-1} en un punto específico. Si un punto aparece más de una vez dentro del vector Ξ, se dice que tiene multiplicidad. Si el primer y el último punto tienen multiplicidad p+1, se dice que el vector de puntos es un *vector abierto* [4].

Un factor importante de una B-Spline formada a partir de un vector de puntos abierto, es que su definición favorece la imposición de condiciones de borde para análisis. En términos geométricos, la B-Spline coincide con sus puntos de control inicial y final [3].

Para definir un sólido B-Spline, se debe considerar una malla de control de nxmxl elementos, ($\boldsymbol{B}_{i,j,k}$), tres vectores de puntos ($\Xi = \xi_1, \dots, \xi_{n+p+1}$, $H = \eta_1, \dots, \eta_{m+q+1}$, $Z = \zeta_1, \dots, \zeta_{l+r+1}$) y tres funciones de base en una dimensión $N_{i,p}$, $M_{j,q}$ y $L_{k,r}$ ($i = 1 \dots n, j = 1 \dots m,\ k = 1 \dots l$). Luego se tiene [4]:

$$\boldsymbol{S}(\xi, \eta, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} N_{i,p}(\xi) M_{j,q}(\eta)\, L_{k,r}(\zeta) \boldsymbol{B}_{i,j,k} \tag{2.3}$$

donde:

ξ: Punto de evaluación en el dominio de Ξ.

η: Punto de evaluación en el dominio de H.

ζ: Punto de valuación en el dominio de Z.

Al construir una NURBS, para cada punto de control se considera, adicionalmente, un peso [5]. Luego, para definir un cuerpo NURBS en tres dimensiones, se utilizan los puntos de control $\boldsymbol{P}_{i,j,k}$, con pesos $w_{i,j,k} > 0$, obteniéndose [3]:

$$\boldsymbol{V}(\xi,\eta,\zeta) = \sum_{1=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l}\boldsymbol{P}_{i,j,k}\,R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) \tag{2.4}$$

Tal que [4]:

$$R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta)L_{k,r}(\zeta)w_{i,j,k}}{\sum_{\hat{\imath}=1}^{n}\sum_{\hat{\jmath}=1}^{m}\sum_{\hat{k}=1}^{l}N_{\hat{\imath},p}(\xi)M_{\hat{\jmath},q}(\eta)L_{\hat{k},r}(\zeta)w_{\hat{\imath},\hat{\jmath},\hat{k}}} \tag{2.5}$$

donde:

p , q, r: Grados polinomiales en direcciones x, y, z, respectivamente.

ξ,η,ζ: Punto de evaluación en el dominio de Ξ, H y Z, respectivamente.

$N_{i,p}(\xi)$, $M_{j,q}(\eta)$, $L_{k,r}(\zeta)$: funciones de forma según ecuaciones 2.1 y 2.2.

$P_{i,j,k}$: punto de control en las coordenadas i, j ,k del dominio de puntos de control en el sólido.

2.3.- Método IGA (IsoGeometric Analysis).

El método de análisis isogeométrico tiene su origen en la búsqueda de la eliminación de errores de cálculo del método de elementos finitos (FEM), provocados por la aproximación de una geometría.

Su principal característica radica en que, sin importar la finesa de la malla utilizada, el método isogeométrico siempre utiliza la geometría real y exacta del cuerpo, tal como su nombre lo indica [6]. Para esta memoria, se utiliza un método basado en funciones tipo NURBS.

Por definición, una" malla" para NURBS se define por los vectores de puntos (por ejemplo de ΞxHxZ, en tres dimensiones), luego los tramos entre los puntos que conforman dichos vectores determinan los elementos. Para definir la geometría, se utilizan los puntos de control, que también se asocian con las funciones de base [6] (ver sección 2.2).

Los coeficientes de las funciones de base definen los grados de libertad, o variables de control. Luego, cualquier cálculo asociado a la geometría (desplazamiento, velocidad, etc.) se representa en términos de estas mismas funciones. Es decir, están isoparametrizadas.

Un arreglo construido en base a parches NURBS, equivalente al subdominio en FEM, pueden ensamblarse en arreglos globales de la misma manera que en método de elemento finito tradicional, solo se debe asegurar la compatibilidad utilizando las mismas definiciones en bordes colindantes [6]. Con esto, el método IGA puede aplicarse para el método de Galerkin, permitiendo su utilización en variados problemas físicos.

2.4.- Modos de vibración de una estructura

Los modos de vibración de una estructura indican las formas en que ésta se mueve cuando es excitada con sus frecuencias naturales. Para cada frecuencia natural se asocia un modo de vibración.

Los modos de vibración y frecuencias naturales se obtienen a partir de la solución de la ecuación característica del sistema, planteada en la Ecuación (2.6), donde K y M pueden obtenerse tanto de manera numérica como analítica [7] (dependiendo de las condiciones de la estructura).

$$(K - p^2 M)\{x\} = 0 \tag{2.6}$$

Para el cálculo numérico, se debe obtener la solución del problema de valor propio que se plantea en la Ecuación (2.6) para los elementos y funciones de forma escogidas en el modelo[7].

Para la obtención experimental de los modos de vibración y frecuencias naturales se debe tener un banco de pruebas donde la estructura se somete a variados rangos de frecuencia. La posición de determinados puntos de la mencionada estructura se mide a partir de métodos ópticos y se determina que una frecuencia dada es una frecuencia natural siempre que genere un modo de vibración constante y se diferencie ampliamente del movimiento fuera de dicha frecuencia [7].

2.5.- Correlación numérica experimental

Para definir la exactitud de un modelo numérico se pueden utilizar diversos métodos que comparan los datos obtenidos por dicho método con datos experimentales. Uno de los métodos más utilizados corresponde a la correlación, aplicando el criterio de *Modal Assurance Criterion (MAC)*, que expresa la correlación que es visualizada en un gráfico de 45° en un solo número. Se define como [7]:

$$MAC_{ij} = \frac{\left(\phi_{a,i}^T \phi_{e,j}\right)^2}{\left(\phi_{a,i}^T \phi_{a,i}\right)\left(\phi_{e,j}^T \phi_{e,j}\right)} \tag{2.20}$$

donde:

$\phi_{a,i}$: i-ésimo modo analítico (o numérico).

$\phi_{e,j}$: j-ésimo modo experimental.

Un valor de 0 indica que no hay correlación, mientras que un valor de 1 indica dos modos perfectamente correlacionados [7].

3.- METODOLOGÍA

La metodología de trabajo se basa en la creación del modelo de elementos finitos, aplicación del mismo al estudio de modos de vibración y comparación de resultados con datos experimentales.

Como ya se ha mencionado, la parametrización de la placa se encuentra delimitada por la continuidad del modelo de estudio con la publicación base para la cual se perfecciona el modelo de elementos finitos, por tanto este parámetro se encuentra fijo en términos de iteraciones o mejoras aplicables al trabajo realizado a lo largo del desarrollo de la memoria.

Finalmente, se desarrollan puntos clave en el trabajo, a saber:

- Entrenamiento en uso de método IGA con NURBS.
- Parametrización de placa compuesta tipo panal de abeja para modelo IGA. Crear programa de elementos finitos para la geometría y condiciones de borde de la placa, se plantea un pseudo-código estándar, que pueda aplicarse a las funciones de forma posteriormente definidas.
- Determinación de funciones de forma, es decir, elección de grados polinomiales y vectores de puntos.
- Aplicación de método IGA en el cálculo de modos de vibración: Consiste iterar el programa, obtener resultados y adaptarlos de manera que sean comparativos con los datos experimentales.
- Comparación de resultado computacional con datos experimentales: Comparación de datos mediante el uso de correlación.
- Concluir.

Como existen muchos tipos de funciones NURBS, puede que se requiera de varias iteraciones de los pasos anteriores, para encontrar las funciones más adecuadas. Para resolver este problema el diagrama de decisión es el que se presenta en la Figura 3.1.

Vale mencionar que el avance desde la etapa 1 a la 2 está determinado por la validación del modelo de placa simple.

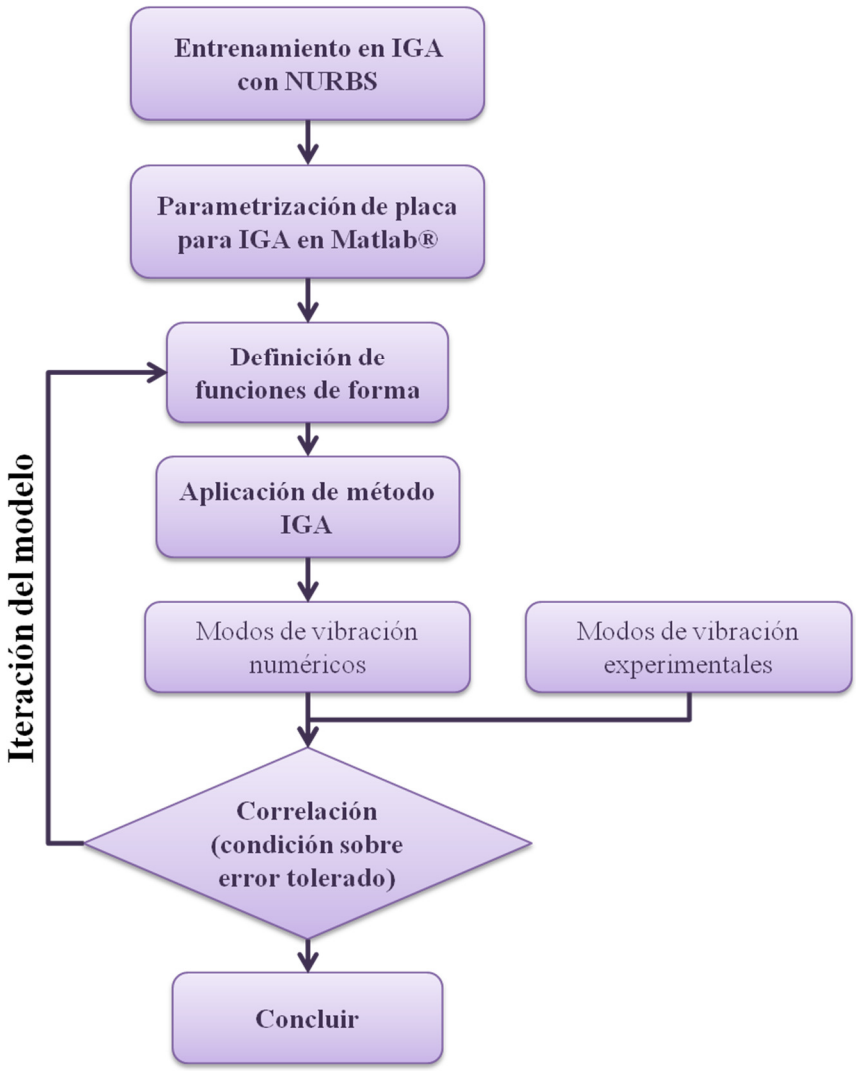


Figura N° 3.1: Metodología de decisión.

4.- ENTRENAMIENTO EN UTILIZACIÓN DE MÉTODO IGA Y NURBS

4.1.- Modelo unidimensional: Modos de pandeo de columna esbelta.

4.1.1.- Implementación computacional de NURBS unidimensionales

Funciones de forma

Para definir la función de forma asociada a un punto arbitrario, u, con u perteneciente a l intervalo [a,b) con a límite inferior del vector de puntos y b límite superior; se debe utilizar la Ecuación (2.4) en una dirección. Esto es:

$$\boldsymbol{C}(u) = \sum_{1=1}^{n} R_i^p(u)\boldsymbol{P}_i \tag{4.1}$$

Con $R_i^p(\xi)$ definida a partir de la Ecuación (2.5) de manera unidimensional como:

$$R_i^p(u) = \frac{N_{i,p}(u)w_i}{\sum_{i=1}^{n} N_{i,p}(u)w_i} \tag{4.2}$$

Donde $N_{i,p}(\xi)$ corresponde a una B-Spline, calculada a partir de la recursión de las ecuaciones 2.1 y 2.2. De dicha recursión, se puede concluir que $N_{i,0}(u)$ es una función escalón cuyo valor es nulo para todo punto fuera del intervalo $[\xi_i, \xi_{i+1})$, por lo cual, como $N_{i,p}(u)$ será definida recursivamente a partir de funciones de grado (p-1), $N_{i,p}(u)$ es nula para cualquier punto fuera del intervalo de interés [8].

De lo anterior, puede concluirse lo siguiente:

- Para encontrar las funciones de base, se debe conocer el intervalo $[\xi_i, \xi_{i+1})$ dentro del vector de puntos al cual el punto de evaluación pertenece. Si se cumple que $\xi_i \leq u < \xi_{i+1}$, entonces se dice que u pertenece al i-ésimo lugar o span.
- Para un determinado span, a lo más p+1 de las funciones de base son no nulas, las que se corresponden a $N_{i-p,p}(u)$ hasta $N_{i,p}(u)$.

Por lo dicho, antes de calcular cualquier función de forma, se debe encontrar el span al que pertenece el punto de evaluación. Para esto, se realiza una búsqueda binaria que entrega como resultado el índice i asociado. Computacionalmente, esto se realiza mediante la función *FindSpan.m*, la cual corresponde a una adaptación para Matlab® del algoritmo A2.1 [8]. El código asociado puede encontrarse en el Anexo 1.

Vale mencionar que los índices calculados mediante *FindSpan.m* comienzan la numeración en cero, lo que debe considerarse al momento de indexar en Matlab®, pues este último numera elementos desde 1.

Para el cálculo de las funciones de base, se asume que u se encuentra en el i-ésimo span y se procede calculando las funciones no nulas en un esquema triangular inverso, esto es:

$$
\begin{array}{ccc}
 & & N_{i-p,p} \\
 & N_{i-1,1} & \\
N_{i,0} & \cdots & \vdots \\
 & N_{i,1} & \\
 & & N_{i,p}
\end{array}
$$

Debe entenderse, entonces, que se computa $N_{i,0}$, luego se avanza hacia la derecha del esquema, obteniéndose sucesivamente las funciones desde p=0 hasta p (grado polinomial deseado) y correspondientes a los puntos de control i-p-ésimmo hasta i-ésimo. Planteando cada término según el algoritmo de las ecuaciones 2.1 y 2.2.

Para ello se utiliza el Algoritmo A2.2 [8], basado en que el segundo término de la Ecuación (2.5) para un determinado i corresponde justamente al primer término de la ecuación para i+1. Con ello, se puede iterar en base a la repetición de dichos términos, con la siguiente notación [8]:

$$left(j) = u - \xi_{i+1-j} \tag{4.3}$$

$$rigth(j) = \xi_{i+j} - u \tag{4.4}$$

$$N_{i-p,p} = \frac{left(p+1)}{rigth(p-1)+left(p+1)} \cdot N_{i-p,p-1} + \frac{left(p-1)}{rigth(p-1)+left(p)} \cdot N_{i-p-1,p-1} \tag{4.5}$$

Para el cálculo de las funciones de base, se crea la función *BasisFuns.m*, que corresponde a la adaptación del algoritmo mencionado para Matlab®, esta puede encontrarse en el Anexo 1. Dicha función entrega un vector con las p+1 funciones de forma asociadas a cada span.

Finalmente, para el cálculo de las funciones de forma NURBS, se tiene que, como solo hay p+1 funciones de forma no nulas asociadas a cada punto, no se requiere de hacer la suma completa sobre el vector de puntos, basta ponderar cada función de forma por el peso asociado al punto de control donde se encuentra y se tiene que:

$$R_i^p = N/sN \tag{4.6}$$

Donde N es el vector de funciones de base ponderado y sN corresponde a la suma de todos los elementos en dicho vector.

Derivadas de funciones de forma

La derivada de una B-Spline viene dada por la Ecuación (4.7), obtenida de las referencias [8], este indica que:

$$N_{i,p}{}' = \frac{p}{\xi_{i+p}-\xi_i} N_{i,p-1}(u) + \frac{p}{\xi_{i+p+1}-\xi_{i+1}} N_{i+1,p-1}(u) \tag{4.7}$$

Para calcular lo anterior, se realiza una adaptación del Algoritmo A2.3 [8] en la función *DerBasisFuns.m*, que se encuentra en el Anexo 1.

Para el caso de la NURBS, la derivada viene dada por la ecuación obtenida de The NURBS Book [9], que dice:

$$R_{i,p}{}' = \frac{A'-w'(u)R_{i,p}(u)}{w(u)} \tag{4.8}$$

Donde A' corresponde a $N_{i,p}{}'$ ponderado por el peso asociado a cada función de base (peso del punto de control que pondera), $w'(u)$ corresponde a la suma de todos los elementos en A' y $w(u)$ es la suma de todos los elementos en $R_{i,p}(u)$.

Computo de puntos en una curva NURBS

Para computar los puntos basta con conocer la ubicación o span del punto de evaluación en el vector de puntos y realizar la operación de multiplicar punto a punto cada función de base con el punto de control correspondiente. Los puntos de control se numeran al igual que las funciones de base, por lo que a un span i le corresponden desde P_{i-p} hasta P_i.

Una adaptación del algoritmo A4.1 [9] con el nombre de *Line_B_Spline.m* se encuentra en el Anexo 1 y permite calcular las funciones de forma asociadas una malla con B-Splines, mientras que su análogo con NURBS se haya en el mencionado anexo bajo el nombre de *Line_NURBS.m*.

4.1.2.- Planteamiento de método IGA

Planteamiento del problema

El problema corresponde a obtener los modos de pandeo de una columna esbelta, los que corresponden a la solución del problema de valor de propio planteado por la ecuación:

$$\begin{bmatrix} EI\left(\frac{d^2y}{dx^2}\right) + Py = 0 \;\; y \in (0, L) \\ y(0) = 0, y(L) = 0 \end{bmatrix} \tag{4.9}$$

Las dimensiones del problema son L=10 [ft] y EI=10^6 [lbf in].

La forma débil del problema se obtiene por método de residuos ponderados, es decir, se pondera la ecuación por un peso arbitrario y luego se integra, de lo que se obtiene que:

$$\int_\Omega [(EIu')' + (-P)u]\omega dx = 0 \tag{4.10}$$

Utilizando la regla de la cadena, se obtiene que:

$$(EIu')'w = \big((EIu')w\big)' - (EIu')w' \tag{4.11}$$

Reemplazando 4.11 en 4.10, se obtiene:

$$\int_\Omega [(EIu'w)' - EIu'\omega' - Pu\omega]dx = 0 \tag{4.12}$$

Por teorema fundamental del cálculo:

$$\int_\Omega (EIu'w)'dx = \mathrm{EIu'(L)\omega(L) - EIu'(0)\omega(0)} \tag{4.13}$$

Por definición de ω, se tiene que ésta tiene valor nulo en los bordes, de manera que la función al ponderarla mantiene las condiciones de contorno, por lo tanto, la expresión de la Ecuación (4.13) es nula. Al incluir este resultado en la Ecuación (4.12) se obtiene la forma débil del problema:

$$\begin{bmatrix} \int_\Omega [EIu'\omega' + Pu\omega]dx = 0 \\ u \in \varrho, ; \varrho = \{u_i : u_i \in H^1(\Omega), u_i(0) = 0\} \\ \omega \in \vartheta; \; \vartheta = \{\omega_i : \omega_i \in H^1(\Omega), \omega_i(0) = 0\} \\ \Omega = (0, L) \end{bmatrix} \tag{4.14}$$

Luego, la solución corresponde a encontrar u y ω tal que 4.14 se cumpla, lo que corresponde a solucionar el problema de valor propio:

$$\begin{cases} B(u,\omega) = l(\omega) \\ B(u,\omega) = \int_\Omega EIu'\omega' dx \\ l(\omega) = \int_\Omega (-P)u\omega dx \end{cases} \tag{4.15}$$

Utilizando el método de Galerkin, sea $u^h = \sum N_{i,p}(u)P_i$, la expresión para un punto u en la parametrización isogeométrica, setiene que $\omega = N_{i,p}$. Si se reemplaza en la Ecuación (4.15), se obtiene:

$$\begin{cases} B\big(N_{i,p}P_i, N_{j,p}\big) = \int_\Omega EI \sum (N_{i,p}P_i)'(N_{j,p})' dx \\ l\big(N_{i,p}P_i, N_{j,p}\big) = \int_\Omega (-P) \sum (N_{i,p}P_i)\,(N_{j,p}) dx \end{cases} \tag{4.16}$$

Como tanto B como l son aplicaciones lineales, se puede dejar la constante P_i fuera de la sumatoria e invertir las operaciones, con lo que queda.

$$B_i\big(N_{j,p}, N_{i,p}\big) = \int_\Omega EI(N_{i,p}P_i)'(N_{j,p})' dx \tag{4.17}$$

$$l_i\big(N_{j,p}, N_{i,p}\big) = \int_\Omega (-P)(N_{i,p}P_i)(N_{j,p}) dx \tag{4.18}$$

Sigue que:

$$\sum B_i\big(N_{j,p}(u), N_{i,p}\big)P_i = \sum l_i\big(N_{j,p}(u), N_{i,p}\big)\,P_i \tag{4.19}$$

Con lo cual se define la matriz de rigidez y el vector de fuerzas:

$$\begin{cases} K_{i,j} = B\big(N_{i,p}, N_{j,p}\big) \\ f_i = l\big(N_{i,p}, N_{j,p}\big) \end{cases} \tag{4.20}$$

La matriz y vector globales se forman por ensamble de la matriz y vector elementales, cada una definida en base a la Ecuación (4.20).

Implementación del método

Para la solución de las integrales, Ecuación (4.20), se debe realizar el cambio de variable desde las coordenadas "reales" (entiéndase coordenadas cartesianas) al sistema isogeométrico (coordenada u), esto es:

$$\int_{x_i}^{x_{i+1}} F(x)dx = \int_{u_i}^{u_{i+1}} F(x(u))|\det(J)|du \tag{4.21}$$

con

$$J = \frac{dx}{du} \tag{4.22}$$

En este caso, F corresponde a las expresiones de la ecuaciones 4.20 (en la integral 4.17) y a las funciones de forma (integral 4.18). En el primer caso, se debe obtener la expresión de las derivadas de primer orden, para lo cual se tiene:

$$\left[\frac{dR}{dx}\right] = J^{-1}\left[\frac{dR}{du}\right] \tag{4.23}$$

En el planteamiento del método IGA, la definición de los vectores de puntos y puntos de control se hace de manera tal que la multiplicidad del primer y último elemento del vector de puntos sea p+1 (p grado del poliniomio). Además, se define la variable *insertionPoints*, que indica el número de puntos que se quieren insertar.

Con fin de realizar un aprendizaje escalonado, primero se plantea la solución utilizando B-Splines, que son funciones más simples y, posteriormente, el programa IGA con NURBS.

En la solución con B-Splines, el problema consiste en encontrar para cada elemento las funciones de forma relacionadas, para ello, por cada elemento se definen los puntos de Gauss para integración y se utiliza *FindSpan.m* para definir el span correspondiente y, en base a ello, se calculan todos los parámetros.

Finalmente, plantean las condiciones de borde y se resuelve. La versión completa de este problema se encuentra en el Anexo 2.1.

Luego, en el planteamiento de IGA con NURBS, la definición de los vectores de puntos y puntos de control se hace de la misma manera que en caso anterior y, además, se crea el vector con los pesos que, en este caso pues la estructura es simple, corresponden a 1.

La versión completa de este problema se encuentra en el Anexo 2.2.

Modos de pandeo calculados.

Dado que el énfasis de esta memoria se encuentra en la implementación de análisis isogeométrico con NURBS, se presentan sólo los resultados de dicho método. Además, El programa FEM que se tiene corresponde a elementos lineales y cuadráticos, con cuatro nodos. Para hacer el análisis comparativo de los resultados, se crean elementos análogos a dicho caso. Los resultados son los siguientes:

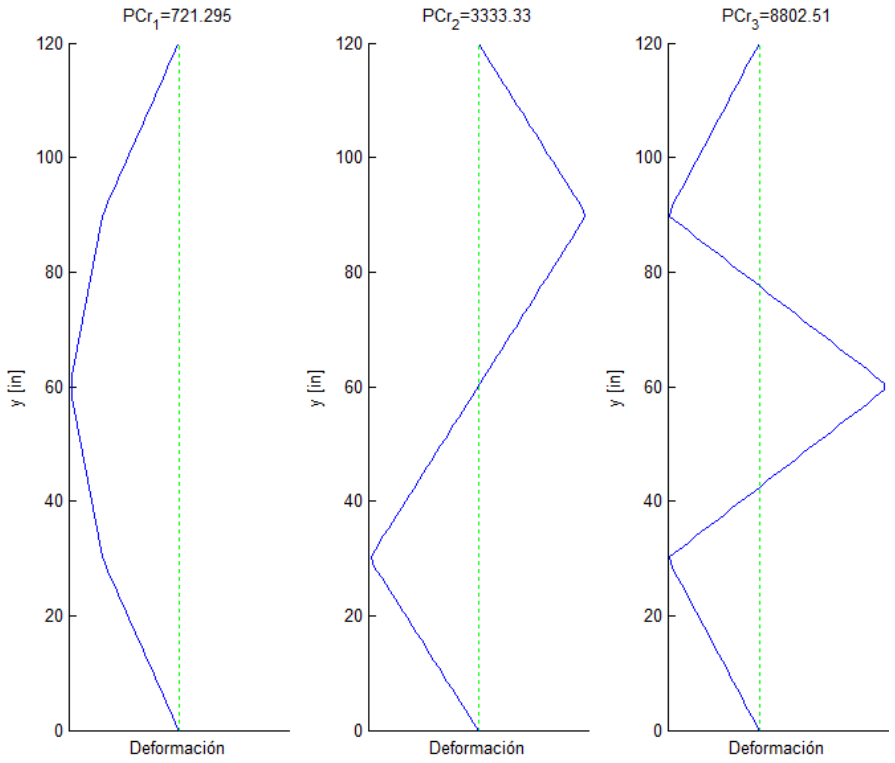


Figura N° 4.1: Modos calculados con NURBS lineales y cuatro elementos.

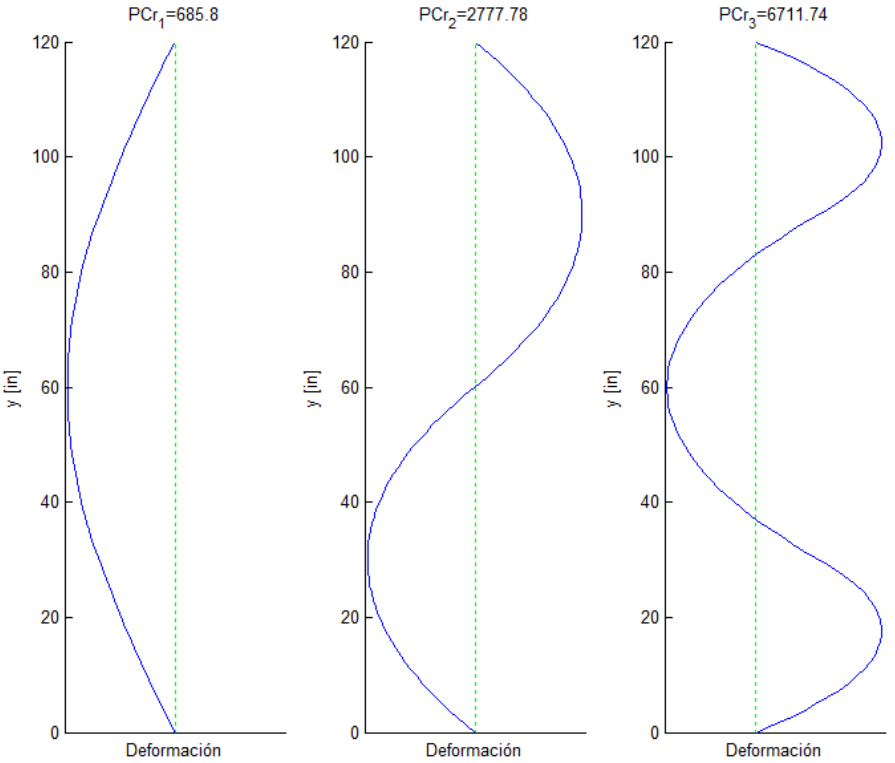


Figura N° 4.2: Modos calculados con NURBS cuadráticos y cuatro elementos.

4.1.3.- Análisis comparativo IGA contra método de elemento finito tradicional (FEM).

El resultado del cálculo con FEM tradicional y elementos lineales se presenta en la Figura 4.3, mientras que el resultado del cálculo con elementos cuadráticos se presenta en la Figura 4.4. Al comparar los gráficos el programa implementado para B-Splines y NURBS se valida con el modelo FEM.

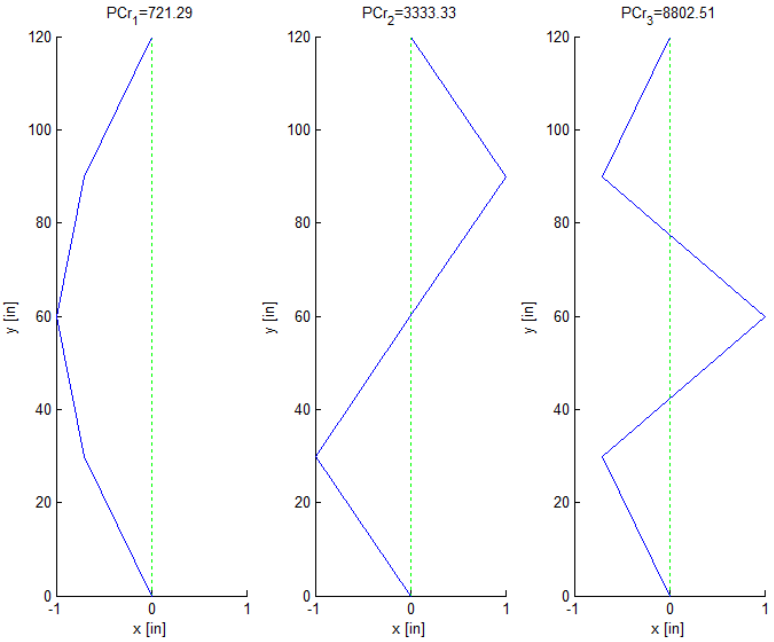


Figura N° 4.3: Modos de pandeo con cuatro elementos lineales

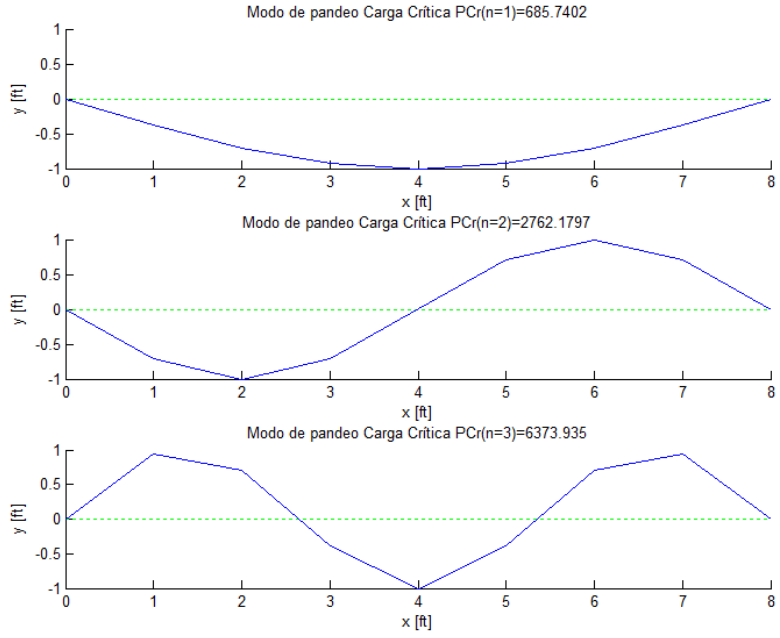


Figura N° 4.4: Modos de pandeo con cuatro elementos cuadráticos.

4.1.4.- Estudio de sensibilidad del modelo

De ahora en más, solo se refiere sobre el programa IGA con NURBS, el cual corresponde al método de interés para esta memoria.

En vista de los resultados y dada la facilidad de aumentar el grado polinomial para los elementos del análisis isogeométrico, se pretende analizar la influencia de esta variación en la exactitud de los resultados. Para efectos comparativos, la Tabla 4.1 contiene las primeras tres cargas críticas calculadas analíticamente.

Tabla N° 4.1: Primeras tres cargas analíticas.

	n=1	n=2	n=3
Carga [lbf]	685,389	2.741,557	6.168,503

Como puede verse en la Figura 4.5, la exactitud del resultado para las dos primeras cargas es mayor que en el caso de FEM (cuadrático) a pesar de utilizar una malla más pequeña. Esto es un indicador de que la influencia en el aumento del grado polinomial es significativa frente a la influencia del mallado para este tipo de problemas. En la Figura 4.6 se muestran las funciones de forma asociadas al análisis mencionado.

Cabe mencionar que la implementación de esta configuración genera una acumulación de deformación hacia los extremos de la viga (ver modo 3 en Figura 4.5). Esto puede asociarse a que las únicas funciones de forma que toman valores unitarios corresponden a aquellas que ponderan al primer y último punto de control, y que las demás funciones toman valores significativamente menores, luego la ponderación hacia los extremos si se utiliza un único elemento genera una visión deformada del resultado.

Para solucionar lo dicho, se plantea que el método debe integrar tanto un aumento del polinomio como de elementos, de manera de distribuir de mejor manera los puntos de control y los valores que ellos ponderan. Puede observarse en la Figura 4.7 que al plantear una solución con polinomio cúbico, bastan 15 elementos para conseguir resultados exactos en las cargas críticas de pandeo. Además, la distribución de las funciones de forma se encuentra en la Figura 4.8, donde puede verse que esta es más homogénea que en el caso anterior.

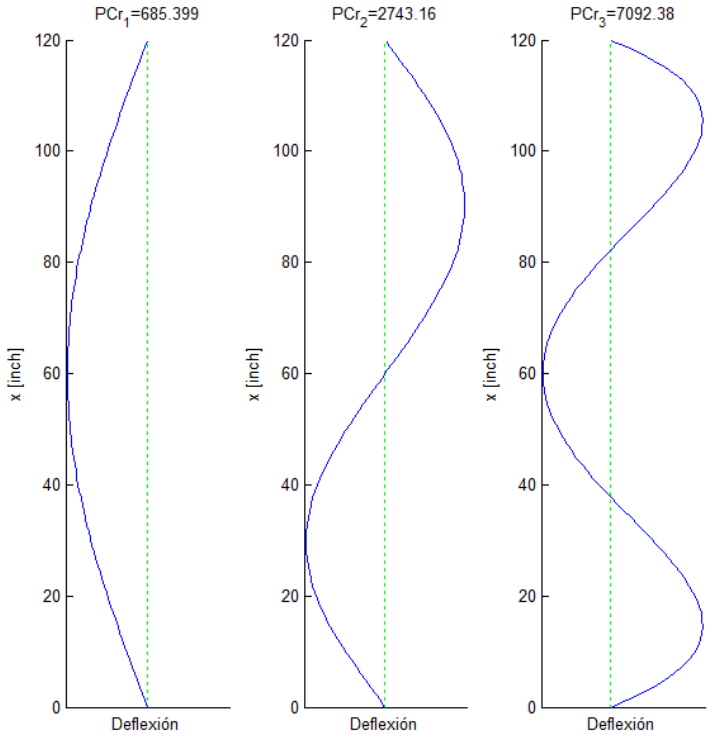


Figura N° 4.5: Pandeo de columna esbelta con polinomios de grado 5 y un elemento.

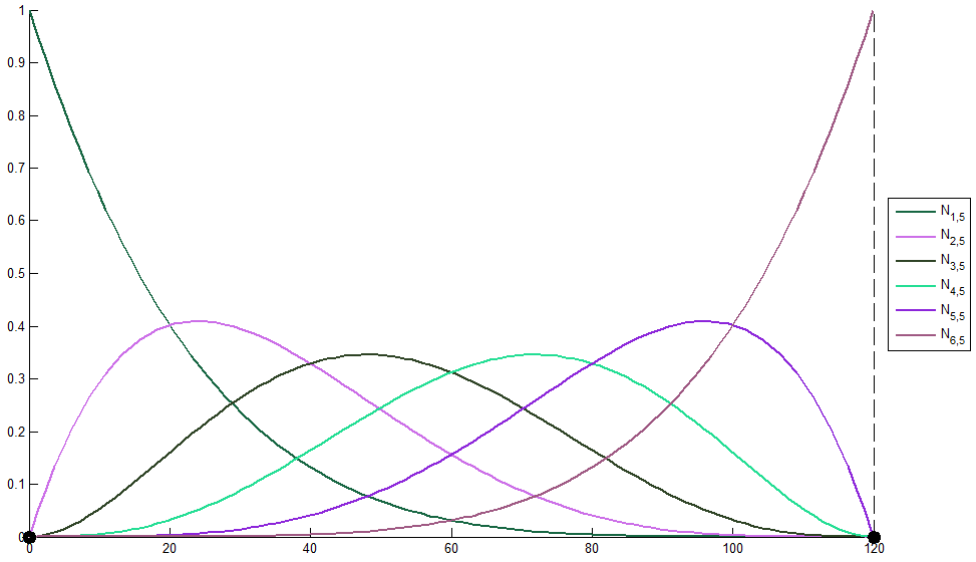


Figura N° 4.6: Funciones de forma para malla de un elemento y polinomio grado 5.

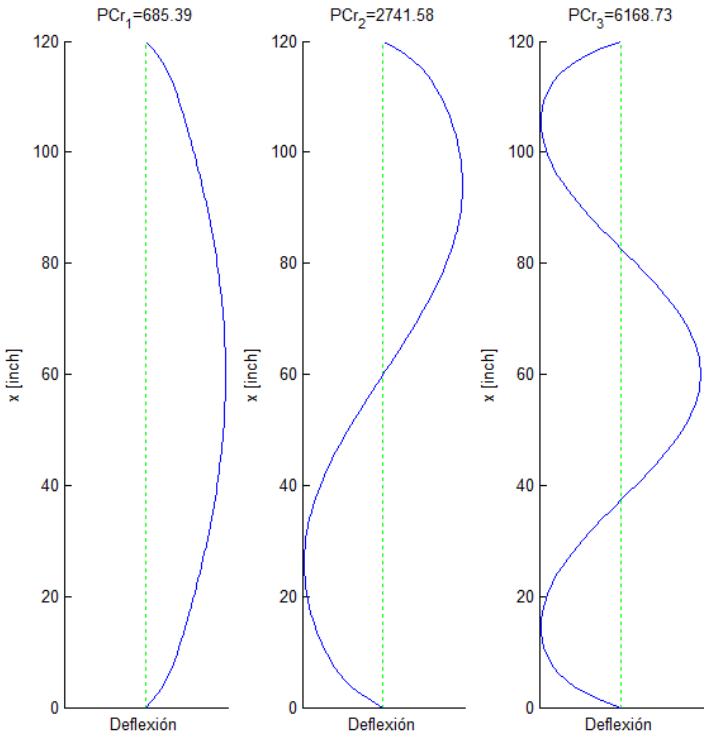


Figura N° 4.7: Pandeo de columna esbelta con polinomios cúbicos y 15 elementos.

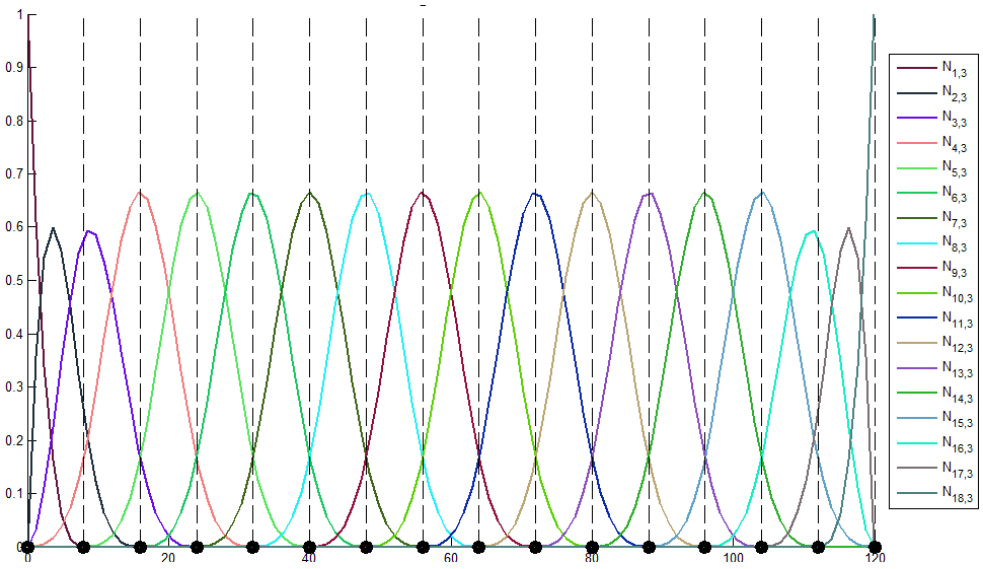


Figura N° 4.8: Funciones de forma para malla de 15 elementos y polinomios cúbicos.

4.2.- Modelo bidimensional: Modos de Vibración de una placa de Kirchhoff.

4.2.1.- Implementación computacional de NURBS bidimensionales

Funciones de forma

Para definir la función de forma asociada a un punto arbitrario se procede de manera análoga al problema unidimensional, utilizando en este caso la definición bidimensional de la Ecuación (2.4). Esto es:

$$\boldsymbol{S}(u,v) = \sum_{j=1}^{m}\sum_{1=1}^{n} R_{i,j}^{p,q}(u,v)\boldsymbol{P}_{i,j} \tag{4.23}$$

Con $R_{i,j}^{p,q}(u,v)$ definida a partir de la Ecuación (2.5) de manera bidimensional como:

$$R_{i,j}^{p,q}(u,v) = \frac{N_{i,p}(u)M_{j,q}(v)w_{i,j}}{\sum_{\hat{\jmath}=1}^{m}\sum_{\hat{\imath}=1}^{n} N_{\hat{\imath},p}(u)M_{\hat{\jmath},q}(v)w_{\hat{\imath},\hat{\jmath}}} \tag{4.24}$$

Para el cómputo de las funciones de forma se utiliza la función unidimensional *BasisFuns.m*, que permite obtener las funciones B-Splines en cada dimensión. Luego, para la malla de un elemento se puede definir la matriz de funciones de forma asociadas a los puntos de control relacionados mediante el producto de los vectores obtenidos mediante *BasisFuns.m*, la ponderación por el peso correspondiente de cada punto de control y la división de la suma de todos los valores del producto vectorial. Es decir:

$$M(v) = \begin{bmatrix} M_{j-q,q}(v) & M_{j-1+1,q} & \cdots\cdots & M_{j-1,q}(v) & M_{j,q}(v) \end{bmatrix} \tag{4.25}$$

$$N(u) = \begin{bmatrix} N_{i-p,p}(u) & N_{i-p+1,p}(u) & \cdots\cdots & N_{i-1,p}(u) & N_{i,p}(u) \end{bmatrix} \tag{4.26}$$

Luego

$$N(u)^T \cdot M(v) = \begin{bmatrix} N_{i-p,p}(u)\cdot M_{j-q,q}(v) & \cdots & N_{i-p,p}(u)\cdot M_{j,q}(v) \\ \vdots & \ddots & \vdots \\ N_{i,p}(u)\cdot M_{j-q,q}(v) & \cdots & N_{i,p}(u)\cdot M_{j,q}(v) \end{bmatrix} \tag{4.27}$$

Y se tiene que:

$$R(u,v) = \left[\frac{1}{\omega(u,v)}\right]\cdot (N(u)^{'}\cdot M(v), W) \tag{4.28}$$

donde $(\cdot\,,\,\cdot)$ representa el producto punto matricial, W es la matriz que contiene los pesos de la malla de puntos de control y $\omega(u,v)$ es la suma de todos los valores del producto vectorial $(N(u)^{'}\cdot M(v), W)$.

Derivadas de funciones de forma

Para el cómputo de las derivadas se utiliza un método similar a la sección anterior, obteniendo las derivadas de manera matricial a partir de la función unidimensional *DerBasisFuns.m*, de la cual se obtiene la derivada direccional en una dimensión.

Para facilitar la lectura, de ahora en más se utiliza la notación R para referirse a $R(u,v)$ y, en general, se obvia la notación de la evaluación en el punto (u,v). Además, el subíndice en cualquier término indica la presencia de una derivada direccional, por ejemplo, R_u nota la derivada de R según u.

Dicho esto, se tiene que [9]:

$$R_u = \frac{A_u - \omega_u \cdot R}{\omega} \tag{4.29}$$

$$R_v = \frac{A_v - \omega_v \cdot R}{\omega} \tag{4.30}$$

$$R_{uu} = \frac{A_{uu} - 2\omega_u R_u - \omega_{uu} R}{\omega} \tag{4.31}$$

$$R_{vv} = \frac{A_{vv} - 2\omega_v R_v - w\omega_{vv} R}{\omega} \tag{4.32}$$

$$R_{uv} = \frac{A_{uv} - \omega_{uv} R - \omega_u R_v - \omega_v R_u}{\omega} \tag{4.33}$$

donde:

A_u: $((N(u)_u)^{\mathrm{T}} \cdot M(v), W)$.

ω_u: Suma de todos los valores de A_u.

A_v: $(\big(N(u)\big)^{\mathrm{T}} \cdot M(v)_v, W)$.

ω_v: Suma de todos los valores de A_v.

A_{uu}: $((N(u)_{uu})^{\mathrm{T}} \cdot M(v), W)$.

ω_{uu}: Suma de todos los valores de A_{uu}.

A_{vv}: $(\big(N(u)\big)^{T} \cdot M(v)_{vv}, W)$.

ω_{vv}: Suma de todos los valores de A_{vv}.

A_u: $((N(u)_u)^{\mathrm{T}} \cdot M(v), W)$.

ω_u: Suma de todos los valores de A_u.

A_{uv}: $((N(u)_u)^{\mathrm{T}} \cdot M(v)_v, W)$.

ω_{uv}: Suma de todos los valores de A_{uv}.

El cómputo de derivadas para B-Splines se realiza con el producto vectorial de las derivadas unidimensionales y constituyen la base para el cálculo de las ecuaciones 4.29 a 4.33.

4.2.2.- Planteamiento de método IGA

Planteamiento del problema a solucionar

Para el planteamiento de la solución de problema de flexión de una placa delgada (placa de Kirchhoff) se busca encontrar la deflexión w, en función de la posición en el plano de la placa. Se considera que el cuerpo se encuentra sometido únicamente a deformación por flexión, por lo que se supone la existencia de un plano neutral en la placa donde no ocurren deformaciones en las direcciones de dicho plano.

Sea una placa con coordenadas cartesianas (x,y,z), la asunción principal asociada a la teoría de placa delgada es que la normal al plano de la placa sin deformar (x-y) se mantiene recta y normal a la superficie neutra de la placa durante la flexión. Esto resulta en que $\gamma_{xz} = \gamma_{yz} = 0$.

Además, se pueden obtener relaciones para los desplazamientos en las direcciones x e y como [10]:

$$\boldsymbol{u} = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{Bmatrix} -z\frac{\delta}{\delta x} \\ -z\frac{\delta}{\delta y} \\ 1 \end{Bmatrix} w = \boldsymbol{L}_u w \tag{4.34}$$

donde:

$\boldsymbol{u}$: Vector de desplazamientos.

u: Desplazamiento en dirección x.

v: Desplazamiento en dirección y.

w: Deflexión.

Luego, el vector de deformaciones queda dado por [10]:

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \frac{\delta u}{\delta x} \\ \frac{\delta v}{\delta y} \\ \frac{\delta u}{\delta y} + \frac{\delta v}{\delta x} \end{Bmatrix} = \begin{Bmatrix} -z\frac{\delta^2 w}{\delta x^2} \\ -z\frac{\delta^2 w}{\delta y^2} \\ -2z\frac{\delta^2 w}{\delta x \delta y} \end{Bmatrix} = -z \begin{Bmatrix} \frac{\delta^2}{\delta x^2} \\ \frac{\delta^2}{\delta y^2} \\ \frac{\delta^2}{\delta x \delta y} \end{Bmatrix} w = -z\boldsymbol{L}_d w \tag{4.35}$$

En particular, para el plano neutro se define la pseudo-deformación según [10]:

$$\boldsymbol{\varepsilon}_p = \boldsymbol{L}_d w \tag{4.36}$$

Planteando la ecuación constitutiva en la Ecuación (4.36) se obtiene el pseudo-esfuerzo como[10]:

$$\boldsymbol{\sigma}_p = \boldsymbol{D}\boldsymbol{\varepsilon}_p \tag{4.37}$$

donde:

$$\boldsymbol{D} = \underbrace{\frac{Eh^3}{12(1-\nu^2)}}_{D} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \tag{4.38}$$

Además, se cumple que:

$$\boldsymbol{\sigma}_p = \begin{Bmatrix} M_{xx} \\ M_{yy} \\ M_{xy} \end{Bmatrix} \tag{4.39}$$

donde:

M_{xx}: Momento flector que genera esfuerzos normales en dirección x (ver Figura 4.9).

M_{yy}: Momento flector que genera esfuerzos normales en dirección y (ver Figura 4.9).

M_{xy}: Momento torsional que genera esfuerzos de corte en planos x-z e y-z (ver Figura 4.9).

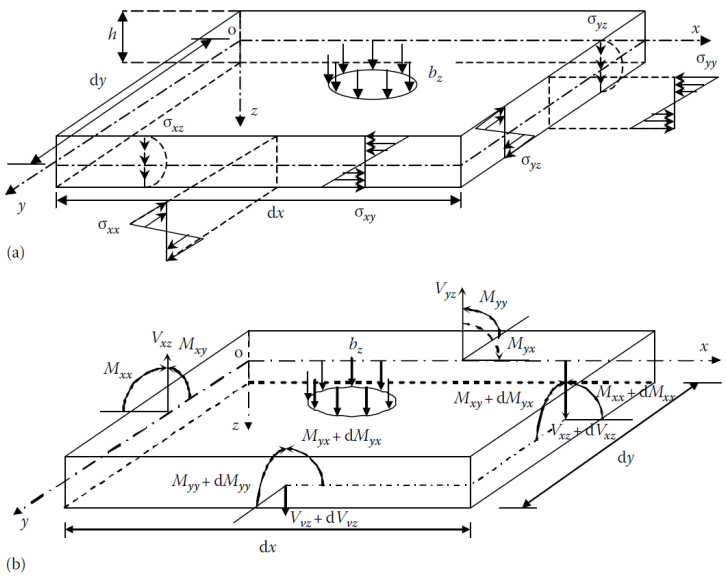


Figura N° 4.9: Celda representativa de dimensiones dx x dy en una placa. (a) Esfuerzos en las secciones. (b) Fuerzas de corte y momentos en las secciones [10].

Al plantear el equilibrio dinámico y ecuaciones constitutivas sobre el estado de cargas mostrado en la Figura 4.9, puede obtenerse la forma fuerte del problema de vibración de una placa de Kirchhoff [10]:

$$D\left(\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4}\right) = b_z \tag{4.40}$$

Para el planteamiento de la forma débil del problema, se considera la teoría utilizada en métodos sin malla, con una discretización del tipo [10]:

$$w^h = \sum_{I \epsilon S_n} \phi_I w_I \tag{4.41}$$

donde:

ϕ_I: Función de forma asociada al I-ésimo punto de control.

w_I: Deflexión del I-ésimo punto de control.

S_n: Conjunto de nodos en el dominio de un punto de interés, en este caso corresponde a los puntos de control asociados a un elemento dado.

A partir del planteamiento del método de Galerkin y tratamientos algebraicos, puede obtenerse la forma débil del problema de vibración libre, dada por [10]:

$$\int_A \delta \boldsymbol{\varepsilon}_p^T \boldsymbol{\sigma}_p dA + \int_\Omega \delta \boldsymbol{u}^T \rho \ddot{\boldsymbol{u}} d\Omega = 0 \tag{4.42}$$

donde:

A: Área de la placa.

Ω: Área del plano neutro.

ρ: Densidad del material.

$\boldsymbol{u}$: Desplazamiento.

$\boldsymbol{\varepsilon}_p$: Pseudo-deformación.

$\boldsymbol{\sigma}_p$: Pseudo-efuerzo.

Luego, si se remplazan las Ecuaciones (4.34), (4.36) y (4.37) en la Ecuación (4.42), se obtiene[11]:

$$\int_\Omega \delta(\boldsymbol{L}_d w)^T \boldsymbol{D}(\boldsymbol{L}_d w)d\Omega + \int_\Omega \delta(\boldsymbol{L}_u w)^T \boldsymbol{D}(\boldsymbol{L}_u w)d\Omega = 0 \tag{4.43}$$

Con lo cual, se puede plantear el problema de valor propio que define los modos y frecuencias naturales como [11]:

$$(K - \omega^2 M)w = 0 \tag{4.44}$$

donde:

M: Matriz de masas.

K: Matriz de rigidez.

w: Vector de deflexiones en los puntos de control.

Por definición [11]:

$$K_{IJ} = \int_\mathrm{A} \boldsymbol{B}_I^T \boldsymbol{D}\boldsymbol{B}_J dA \tag{4.45}$$

con:

$$\boldsymbol{B}_I = \boldsymbol{L}_d \phi_I = \begin{Bmatrix} -\phi_{I,xx} \\ -\phi_{I,yy} \\ -2\phi_{I,xy} \end{Bmatrix} \tag{4.46}$$

y A el área del elemento, además se tiene que:

$$M_{IJ} = \int_A \rho \phi_I \phi_J h dA \tag{4.47}$$

Cabe mencionar que esta solución corresponde a una aproximación de un grado de libertad (deflexión) sin rotación. Además, la matriz de masa corresponde a una aproximación para la cual se descartan términos de órdenes mayores a dos en el espesor (h), pues se considera una placa delgada.

Implementación del método

Dada la definición del problema, es importante considerar que cualquier función de forma planteada debe tener como mínimo grado cúbico. Dicho esto, para la solución de las integrales (ecuaciones 4.45 y 4.47) se debe realizar el cambio de variable, extendiendo la expresión 4.21 a dos dimensiones (cartesianas x,y a coordenadas isogeométricas u,v), se tiene:

$$\int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} F(x,y) dx dy = \int_{u_i}^{u_{i+1}} \int_{v_j}^{v_{j+1}} F(x(u), y(u)) |\det(J)| du dv \tag{4.48}$$

con

$$J = \begin{bmatrix} \frac{dx}{du} & \frac{dy}{du} \\ \frac{dx}{dv} & \frac{dy}{dv} \end{bmatrix} \tag{4.49}$$

En este caso, F corresponde a las expresiones de la ecuaciones 4.46 (en la integral 4.45) y a las funciones de forma (integral 4.47). En el primer caso, se debe obtener la expresión de las derivadas direccionales de segundo orden y de primer orden (cruzadas), para lo cual se tiene:

$$\begin{bmatrix} \frac{d^2R(u,v)}{dx^2} & \frac{d^2R(u,v)}{dydx} \\ \frac{d^2R(u,v)}{dydx} & \frac{d^2R(u,v)}{dy^2} \end{bmatrix} = \begin{bmatrix} \frac{d}{dx} \\ \frac{d}{dy} \end{bmatrix} \cdot \begin{bmatrix} \frac{dR(u,v)}{dx} & \frac{dR(u,v)}{dy} \end{bmatrix} \tag{4.50}$$

Por otro lado:

$$\begin{bmatrix} \frac{d}{dx} \\ \frac{d}{dy} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{d}{du} \\ \frac{d}{dv} \end{bmatrix} \tag{4.51}$$

$$\begin{bmatrix} \frac{dR(u,v)}{dx} & \frac{dR(u,v)}{dy} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{dR(u,v)}{du} \\ \frac{dR(u,v)}{dv} \end{bmatrix} = \begin{bmatrix} \frac{dR(u,v)}{du} & \frac{dR(u,v)}{dv} \end{bmatrix} J^{-1} \tag{4.52}$$

Luego

$$\begin{bmatrix} \frac{d^2R(u,v)}{dx^2} & \frac{d^2R(u,v)}{dydx} \\ \frac{d^2R(u,v)}{dydx} & \frac{d^2R(u,v)}{dy^2} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{d}{du} \\ \frac{d}{dv} \end{bmatrix} \begin{bmatrix} \frac{dR(u,v)}{du} & \frac{dR(u,v)}{dv} \end{bmatrix} J^{-1} \tag{4.53}$$

$$\begin{bmatrix} \frac{d^2R(u,v)}{dx^2} & \frac{d^2R(u,v)}{dydx} \\ \frac{d^2R(u,v)}{dydx} & \frac{d^2R(u,v)}{dy^2} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{d^2R(u,v)}{du^2} & \frac{d^2R(u,v)}{dudv} \\ \frac{d^2R(u,v)}{dudv} & \frac{d^2R(u,v)}{dv^2} \end{bmatrix} J^{-1} \tag{4.54}$$

Finalmente, se plantean las soluciones de la misma manera que en el caso unidimensional. Primero un programa con B-Splines y luego se complejiza para NURBS. Ambos casos se presentan a continuación.

La definición de los vectores de puntos y puntos de control es análoga al caso unidimensional, duplicando las variables. Se identifica cada variable con la dirección espacial asociada, de manera de variar parámetros de manera independiente.

Para implementar B-Splines, se definen entonces los intervalos de integración de acuerdo al elemento, procediendo de manera símil al caso unidimensional en cada dirección y se aplican las relaciones definidas en las secciones anteriores del capítulo 4.2.

La versión completa de este problema se encuentra en el Anexo 3.1.

Finalmente, en la implementación con NURBS, se procede de manera análoga al caso anterior, planteando las funciones de forma correspondientes. Cabe mencionar que se tiene especial cuidado en generar de esta solución una función estándar que permita obtener las matrices de rigidez y masa para una placa cualquiera en función de sus dimensiones y propiedades, tal que pueda ser utilizada en la obtención de modos del panel compuesto.

La versión completa de este problema se encuentra en el Anexo 3.2.

4.2.3.- Análisis comparativo IGA contra modos calculados en Mechanical APDL de ANSYS®.

Dado que el énfasis de esta memoria se encuentra en la implementación de análisis isogeométrico con NURBS, a continuación se presentan sólo los resultados de dicho método.

Las dimensiones y propiedades de la placa utilizada en esta sección corresponden a las características de las placas exteriores del panel compuesto objetivo de este trabajo, las que se presentan en la Tabla 4.2.

Tabla N° 4.2: Propiedades de placa exterior.

Propiedad	
Largo (x)	0,35 [m]
Ancho (y)	0,25 [m]
Espesor	8 E^{-4} [m]
Módulo de Young	6,1372 E^{10} [Pa]
Módulo de Poisson	0,33
Densidad	2,5279 E^{3} [kg/m^3]

Los modos de vibración obtenidos se relacionan con los modos de FEM (ANSYS®), pero es relevante mencionar que las frecuencias no están igualmente correlacionadas. Esto es, en el orden que se ha calculado mediante análisis isogeométrico no se corresponden los modos con el orden en que Mechanical APDL entrega los resultados. Para efectos comparativos, en la tabla siguiente se han ordenado los datos de acuerdo a la correspondencia de los modos de FEM.

Tabla N° 4.3: Frecuencias naturales calculadas.

Mechanical APDL	p,q= 3	p,q= 5	p,q= 7
28,83	27,32	26,44	25,87
33,17	33,87	35,34	37,25
67,83	64,73	63,64	63,63
68,31	70,74	75,60	81,69
84,63	84,52	87,61	92,11
99,78	101,48	106,66	113,33
128,67	125,84	128,09	132,45
141,44	135,43	135,36	138,19
187,26	195,76	211,92	-
189,09	194,75	207,44	220,36
208,16	213,21	228,40	244,70
215,72	205,97	206,36	211,27

Cabe mencionar que tanto para FEM como para IGA se plantea una malla de 35x25 elementos. En el primer modelo se utilizan elementos de tipo *shell* con cuatro nodos, mientras que en el segundo se prueban tres grados polinomiales diferentes.

De acuerdo a los resultados, se calcula el error relativo en las frecuencias para cada modo, con lo cual se obtiene que el modelo IGA que más se asemeja al modelo FEM es aquel con

polinomio cúbico (ver Tabla 4.4). Esto es esperable puesto que los elementos tipo *shell* de cuatro nodos utilizan funciones de forma del mismo orden.

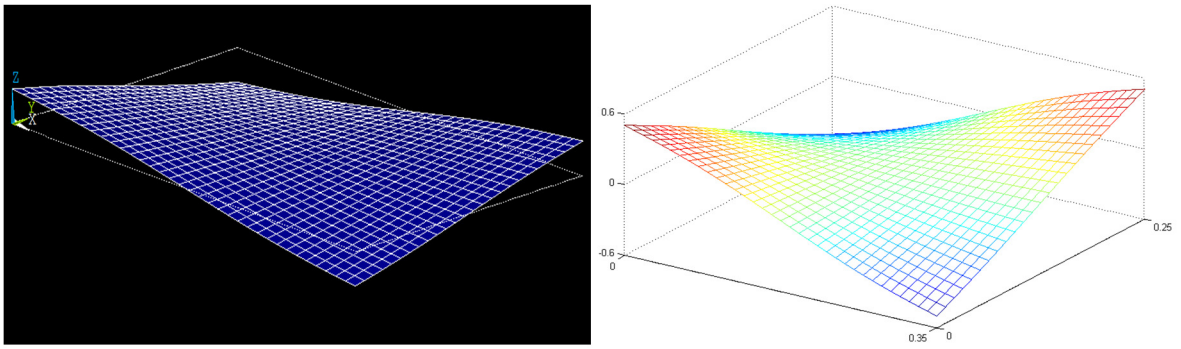
Tabla N° 4.4: Error porcentual de frecuencias entre modos calculados con FEM e IGA.

	p,q= 3	p,q= 5	p,q= 7
Modo 1	5,2%	8,3%	10,3%
Modo 2	2,1%	6,5%	12,3%
Modo 3	4,6%	6,2%	6,2%
Modo 4	3,6%	10,7%	19,6%
Modo 5	0,1%	3,5%	8,8%
Modo 6	1,7%	6,9%	13,6%
Modo 7	2,2%	0,5%	2,9%
Modo 8	4,3%	4,3%	2,3%
Modo 9	4,5%	13,2%	-
Modo 10	3,0%	9,7%	16,5%
Modo 11	2,4%	9,7%	17,6%
Modo 12	4,5%	4,3%	2,1%

Al aumentar el grado del polinomio la respuesta del modelo IGA diverge respecto del modelo comparativo. Pese a lo anterior, la cercanía de resultados para modelos comparables (polinomio cúbico) sirve como validación del método isogeométrico.

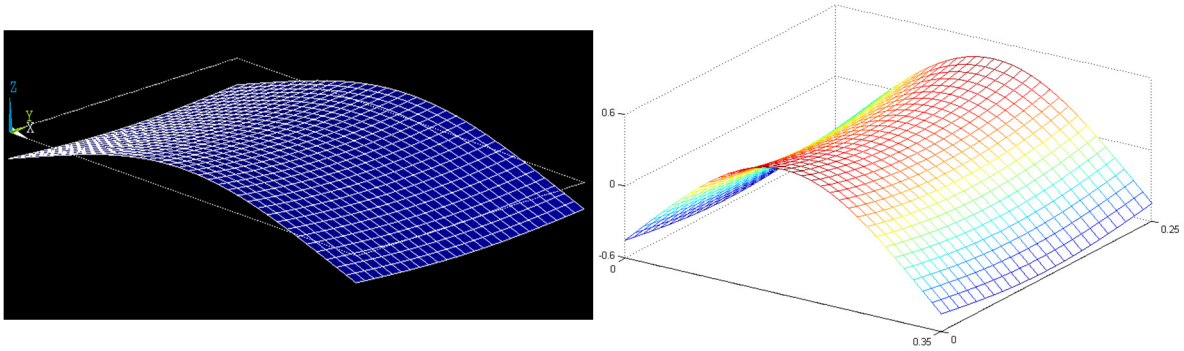
Dicho lo anterior, aunque no es posible discernir sobre la precisión de los métodos, la cercanía de resultados para modelos comparables (polinomio cúbico) sirve como validación del método isogeométrico.

A continuación, se presentan de manera comparativa los modos obtenidos tanto en Mechanical APDL como en IGA, para el mallado de polinomio de orden 3.



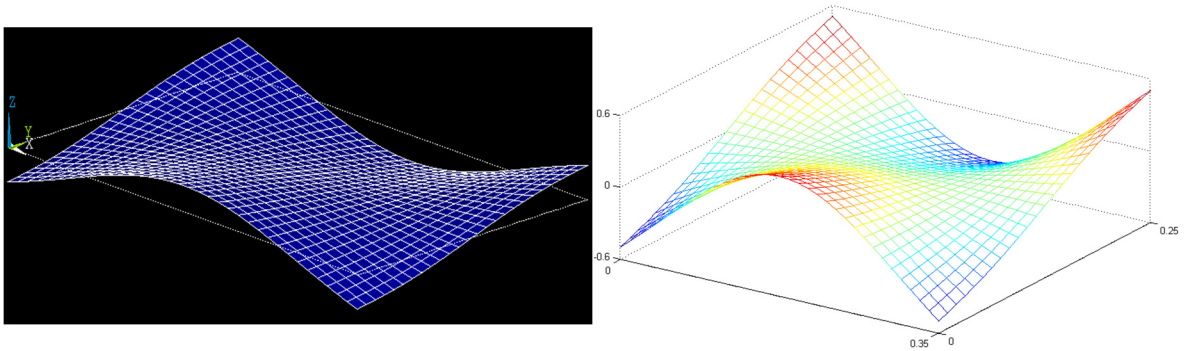
(a) (b)

Figura N° 4.10: Comparación de modo N°1. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



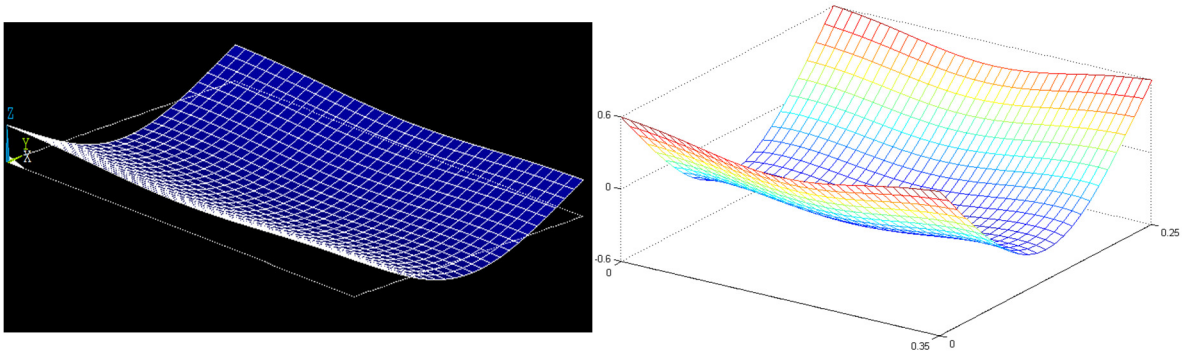
(a) (b)

Figura N° 4.11: Comparación de modo N°2. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



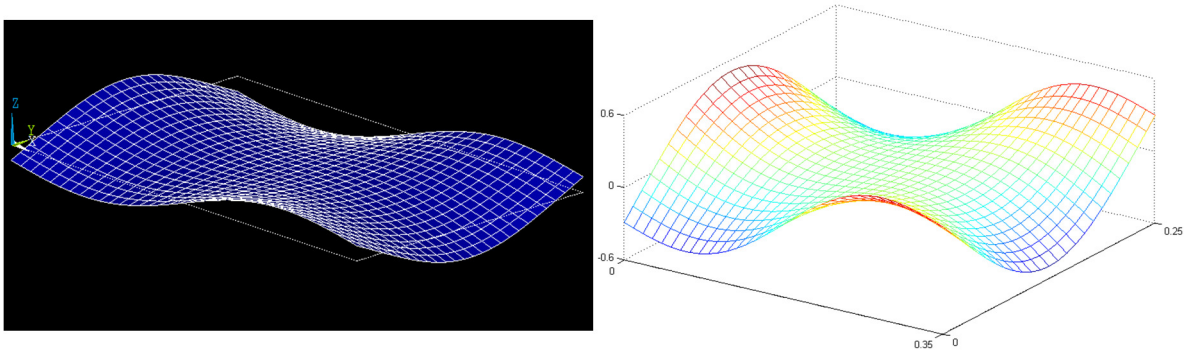
(a) (b)

Figura N° 4.12: Comparación de modo N°3. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



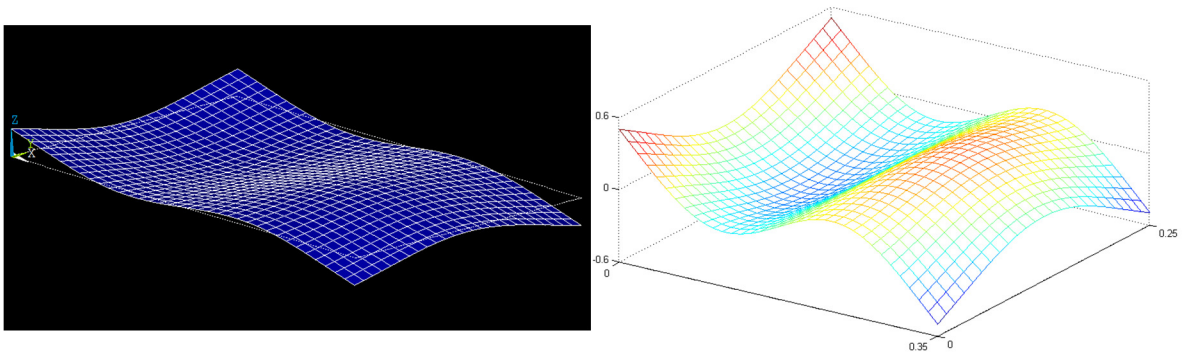
(a) (b)

Figura N° 4.13: Comparación de modo N°4. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



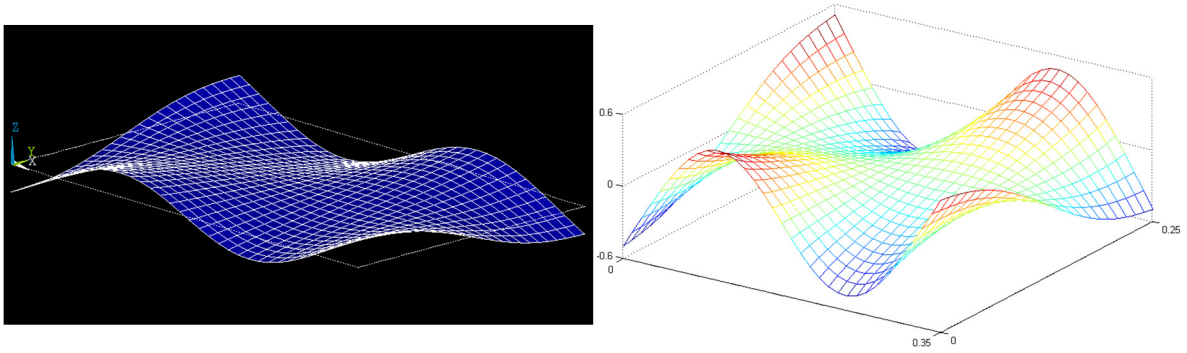
(a) (b)

Figura N° 4.14: Comparación de modo N°5. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



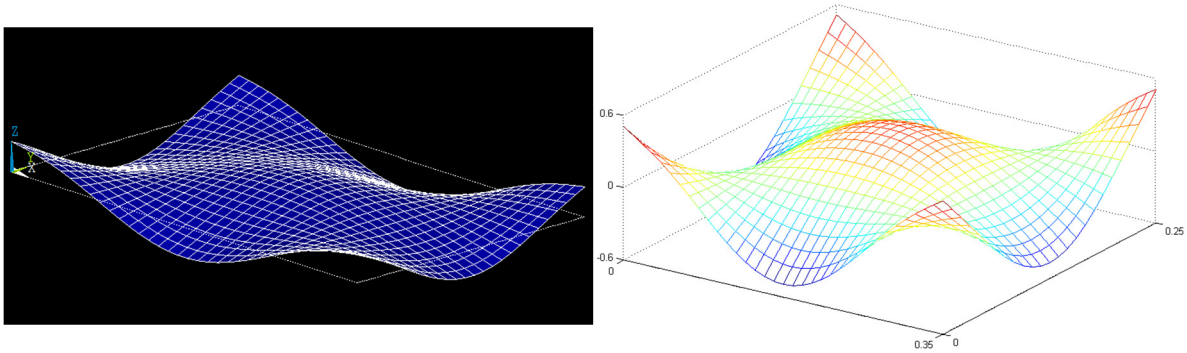
(a) (b)

Figura N° 4.15: Comparación de modo N°6. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



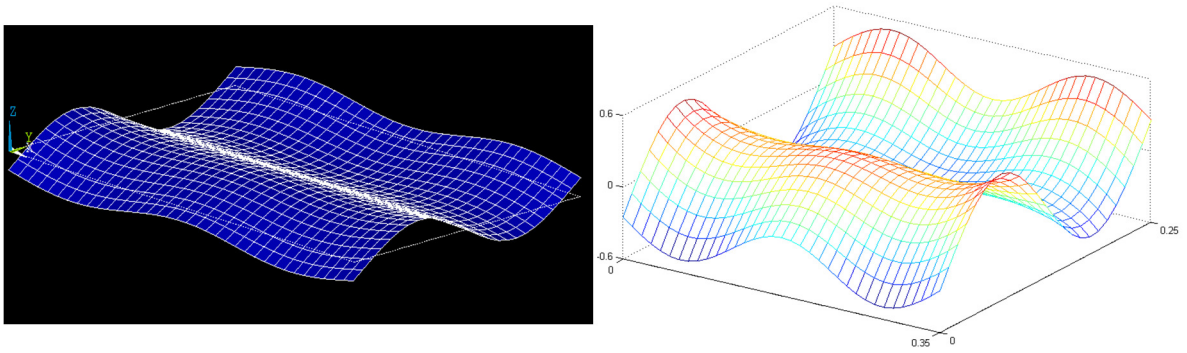
(a) (b)

Figura N° 4.16: Comparación de modo N°7. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



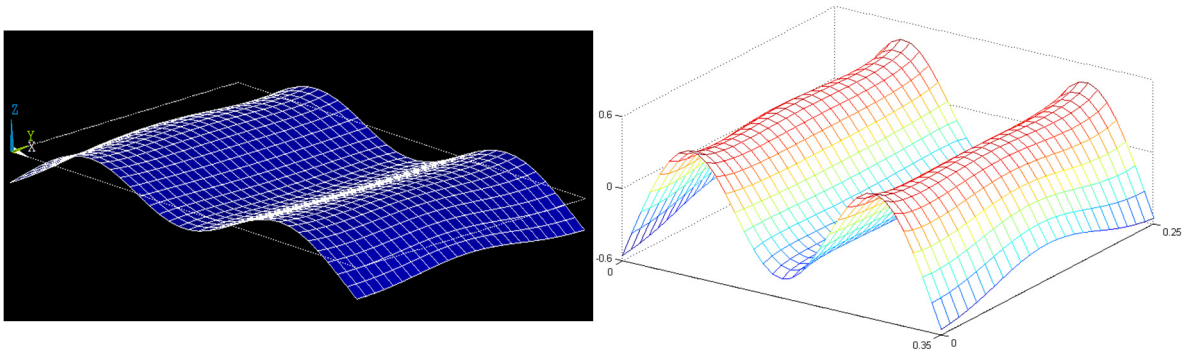
(a) (b)

Figura N° 4.17: Comparación de modo N°8. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



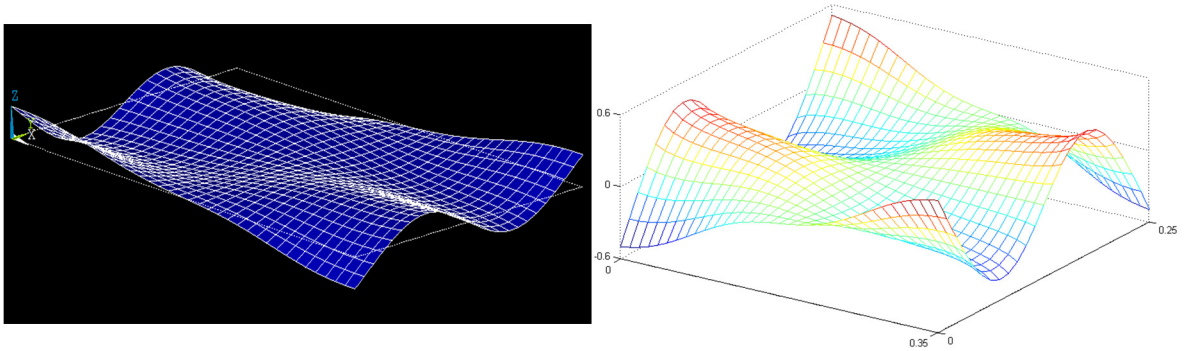
(a) (b)

Figura N° 4.18: Comparación de modo N°9. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



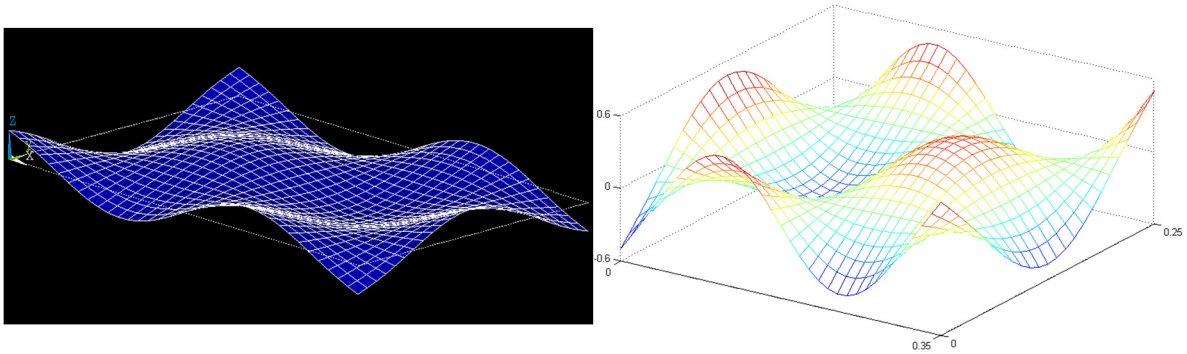
(a) (b)

Figura N° 4.19: Comparación de modo N°10. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



(a) (b)

Figura N° 4.20: Comparación de modo N°11. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.



(a) (b)

Figura N° 4.21: Comparación de modo N°12. (a) Mechanical APDL. (b) IGA con NURBS cúbicas.

5.- MODOS DE VIBRACIÓN PANEL COMPUESTO TIPO PANAL DE ABEJA

5.1.- Planteamiento del modelo numérico.

Para el modelo del panel compuesto, se utiliza la función *Bending_plate_NURBS.m* creada en la sección 4.2, de la cual se obtienen las matrices de rigidez y masa de las placas exteriores y del modelo de placa del núcleo.

Para la unión entre los diferentes niveles, se agregan elementos de resorte, los cuales unen nodo a nodo las placas exteriores con el centro.

Las propiedades de las placas exteriores se presentan en la Tabla 4.2 (ver sección 4.2) mientras que las características del centro se muestran en la Tabla 5.1. Las rigideces de los resortes se consideran de k=8,2584 E^3 [N/m].

Tabla N° 5.1: Propiedades del núcleo.

Propiedad	
Largo (x)	0,35 [m]
Ancho (y)	0,25 [m]
Espesor	10 E^{-3} [m]
Módulo de Young	2,4113 E^{10} [Pa]
Módulo de Poisson	0,33
Densidad	31,1044 [kg/m^3]

El programa creado permite obtener los modos de vibración, frecuencias naturales y parámetros relevantes del modelo como un archivo ".mat", el cual es estudiado posteriormente. Se procede de este modo para optimizar el tiempo de trabajo, es decir, primero se realizan todas las iteraciones y luego se analizan los datos obtenidos. El mencionado programa se puede encontrar en el anexo 4.1 bajo el título de *Honneyvomb.m*.

5.2.- Análisis de comportamiento del modelo.

5.2.1.- Modos y frecuencias calculadas.

Para el análisis de comportamiento del modelo, se utiliza la función *Comparar_modos_EXP.m*, la cual calcula la correlación MAC numérica y experimental y permite graficar tanto en paralelo como sobrepuestos los modos. Cabe mencionar que para efectos visuales, algunos de los modos numéricos se grafican de manera invertida, lo que no genera pérdida de generalidad pues la vibración es simétrica respecto del plano neutro.

Además, como los modos experimentales fueron medidos sobre la placa exterior superior, solo se calcula la correlación en los puntos asociados a dicha estructura.

Cabe mencionar que las frecuencias numéricamente calculadas se han ordenado de acuerdo a la correspondencia de los modos de vibración, por lo cual la numeración asignada no se corresponde necesariamente con aquella obtenida directamente del modelo. En particular, esto sucede para los modos 5 y 6, que numéricamente corresponden a los modos 6 y 7.

A continuación, se presentan las frecuencias experimentales, seguidas de los resultados para diversos ordenes polinomiales en tres tamaños de malla.

Tabla N° 5.2: Frecuencias naturales experimentales de panel compuesto.

	Frecuencia [Hz]
Modo 1	483,07
Modo 2	621,02
Modo 3	986,67
Modo 4	1214,60
Modo 5	1377,00
Modo 6	1528,90

Tabla N° 5.3: Frecuencias [Hz] calculadas para malla de 35x25 elementos.

	p,q= 3	**p,q= 5**	**p,q= 8**	**p,q= 10**	**p,q= 12**
Modo 1	474,46	465,28	457,83	455,68	455,04
Modo 2	563,19	589,63	639,55	676,76	715,35
Modo 3	853,55	872,14	907,86	935,94	966,24
Modo 4	883,26	939,65	1.035,39	1.101,15	1.165,13
Modo 5	992,55	1.052,60	1.149,63	1.214,79	1.277,58
Modo 6	1.058,16	1.116,09	1.207,17	1.268,38	1.327,76

Tabla N° 5.4: Error relativo porcentual de frecuencias numéricas para malla de 35x25 elementos.

	p,q= 3	**p,q= 5**	**p,q= 8**	**p,q= 10**	**p,q= 12**
Modo 1	1,8%	3,7%	5,2%	5,7%	5,8%
Modo 2	9,3%	5,1%	3,0%	9,0%	15,2%
Modo 3	13,5%	11,6%	8,0%	5,1%	2,1%
Modo 4	27,3%	22,6%	14,8%	9,3%	4,1%
Modo 5	27,9%	23,6%	16,5%	11,8%	7,2%

Modo 6	30,8%	27,0%	21,0%	17,0%	13,2%

Tabla N° 5.5: Correlación de modos para malla de 35x25 elementos.

	p,q= 3	p,q= 5	p,q= 8	p,q= 10	p,q= 12
Modo 1	0,998	0,998	0,997	0,997	0,996
Modo 2	0,963	0,982	0,994	0,995	0,992
Modo 3	0,940	0,974	0,994	0,996	0,991
Modo 4	0,881	0,937	0,978	0,989	0,991
Modo 5	0,635	0,663	0,678	0,691	0,710
Modo 6	0,779	0,872	0,938	0,958	0,965

Tabla N° 5.6: Frecuencias [Hz] calculadas para malla de 50x30 elementos.

	p,q= 3	p,q= 5	p,q= 8	p,q= 10	p,q= 12	p,q= 15	p,q= 19
Modo 1	493,58	483,04	472,92	468,79	466,14	464,17	463,95
Modo 2	584,36	600,42	632,35	657,27	684,20	727,05	785,86
Modo 3	958,08	963,62	979,64	994,77	1.012,74	1.043,61	1.088,98
Modo 4	995,72	1.045,31	1.134,40	1.198,28	1.262,43	1.354,34	1.460,91
Modo 5	1.170,54	1.219,57	1.302,50	1.361,33	1.420,97	1.508,51	1.614,65
Modo 6	1.272,05	1.317,88	1.392,55	1.445,24	1.498,82	1.578,04	1.675,76

Tabla N° 5.7: Error relativo porcentual de frecuencias numéricas para malla de 50x30 elementos.

	p,q= 3	p,q= 5	p,q= 8	p,q= 10	p,q= 12	p,q= 15	p,q= 19
Modo 1	2,2%	0,0%	2,1%	3,0%	3,5%	3,9%	4,0%
Modo 2	5,9%	3,3%	1,8%	5,8%	10,2%	17,1%	26,5%
Modo 3	2,9%	2,3%	0,7%	0,8%	2,6%	5,8%	10,4%
Modo 4	18,0%	13,9%	6,6%	1,3%	3,9%	11,5%	20,3%
Modo 5	15,0%	11,4%	5,4%	1,1%	3,2%	9,6%	17,3%
Modo 6	16,8%	13,8%	8,9%	5,5%	2,0%	3,2%	9,6%

Tabla N° 5.8: Correlación de modos para malla de 50x30 elementos.

	p,q= 3	p,q= 5	p,q= 8	p,q= 10	p,q= 12	p,q= 15	p,q= 19
Modo 1	0,998	0,998	0,998	0,997	0,997	0,996	0,996
Modo 2	0,955	0,972	0,988	0,994	0,996	0,994	0,987

Modo 3	0,939	0,964	0,986	0,993	0,995	0,992	0,982
Modo 4	0,874	0,925	0,969	0,983	0,989	0,990	0,982
Modo 5	0,630	0,678	0,722	0,741	0,756	0,775	0,792
Modo 6	0,806	0,874	0,931	0,952	0,962	0,967	0,959

Tabla N° 5.9: Frecuencias [Hz] calculadas para malla de 70x50 elementos.

	p,q= 3	**p,q= 5**	**p,q= 8**	**p,q= 10**	**p,q= 12**	**p,q= 19**
Modo 1	514,95	505,00	493,62	487,82	483,16	473,35
Modo 2	606,57	615,37	634,32	649,83	667,10	737,05
Modo 3	1.090,79	1.083,61	1.080,64	1.082,97	1.088,20	1.123,47
Modo 4	1.126,00	1.153,55	1.207,93	1.250,39	1.296,29	1.469,72
Modo 5	1.437,09	1.468,33	1.526,90	1.571,58	1.619,24	1.793,17
Modo 6	1.630,75	1.655,62	1.703,02	1.739,86	1.779,64	1.927,63

Tabla N° 5.10: Error relativo porcentual de frecuencias numéricas para malla de 70x50 elementos.

	p,q= 3	**p,q= 5**	**p,q= 8**	**p,q= 10**	**p,q= 12**	**p,q= 19**
Modo 1	6,6%	4,5%	2,2%	1,0%	0,0%	2,0%
Modo 2	2,3%	0,9%	2,1%	4,6%	7,4%	18,7%
Modo 3	10,6%	9,8%	9,5%	9,8%	10,3%	13,9%
Modo 4	7,3%	5,0%	0,5%	2,9%	6,7%	21,0%
Modo 5	4,4%	6,6%	10,9%	14,1%	17,6%	30,2%
Modo 6	6,7%	8,3%	11,4%	13,8%	16,4%	26,1%

Tabla N° 5.11: Correlación de modos para malla de 70x50 elementos.

	p,q= 3	**p,q= 5**	**p,q= 8**	**p,q= 10**	**p,q= 12**	**p,q= 19**
Modo 1	0,998	0,998	0,998	0,998	0,997	0,996
Modo 2	0,950	0,963	0,979	0,986	0,991	0,995
Modo 3	0,936	0,954	0,974	0,983	0,989	0,993
Modo 4	0,846	0,884	0,928	0,949	0,964	0,985
Modo 5	0,593	0,623	0,642	0,646	0,646	0,650
Modo 6	0,830	0,873	0,917	0,936	0,949	0,962

5.2.2.- Análisis de comportamiento frente a variación de número de elementos.

De acuerdo a los resultados de la sección anterior, se realiza la agrupación de datos para analizar el efecto del tamaño de la malla en la exactitud de los modos. Los gráficos siguientes muestran la correlación de los seis modos comparados para las tres mallas estudiadas.

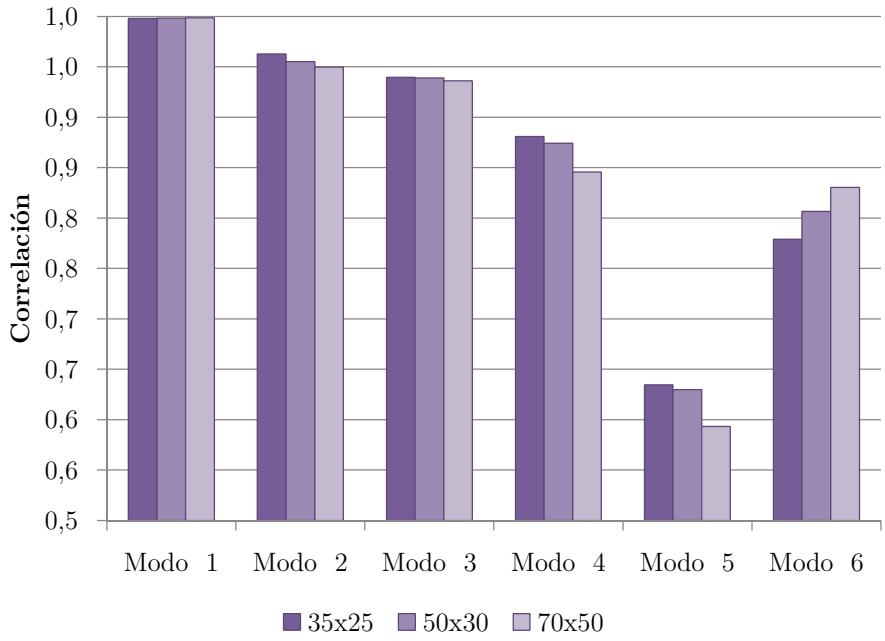


Gráfico N° 5.1: Correlación de modos experimentales y numéricos según mallado para funciones de forma cúbicas.

De acuerdo al Gráfico 5.1, el aumento del tamaño de malla no genera una diferencia significativa en cuanto a la precisión de los modos obtenidos. No obstante, si se observan los gráficos 5.2 y 5.3, puede verse que para funciones de forma de órdenes superiores el aumento del número de elementos tiene una influencia cada vez mayor.

Además, resulta de este estudio que la cantidad de elementos no es directamente proporcional a un aumento de la precisión, sino que incluso para la malla más fina la correlación disminuye (en la mayoría de los modos).

Finalmente, se determina que elementos cuadrados (casos de malla de 35x25 y 70x50) generan resultados menos exactos que los elementos rectangulares (malla de 50x30).

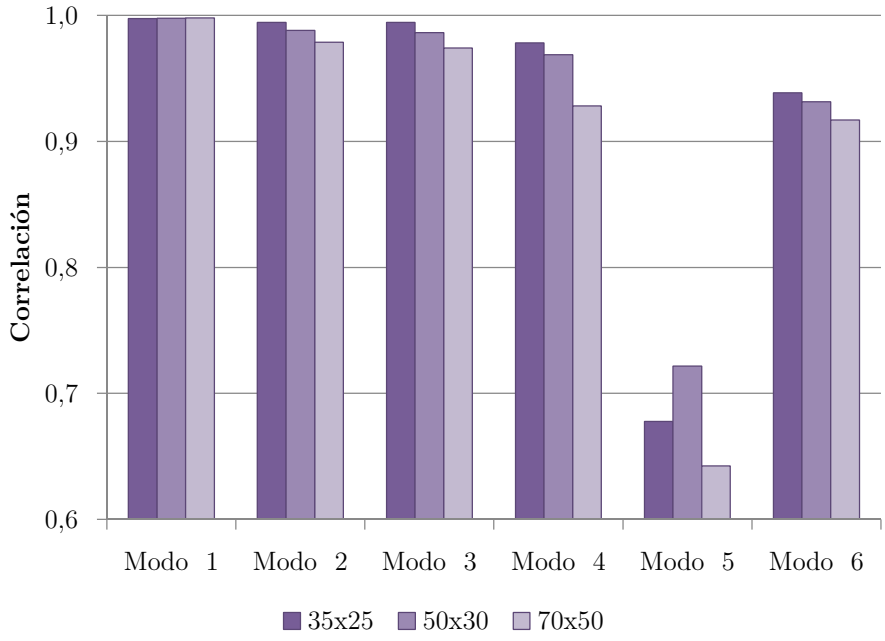


Gráfico N° 5.2: Correlación de modos experimentales y numéricos según mallado para funciones de forma de grado 8.

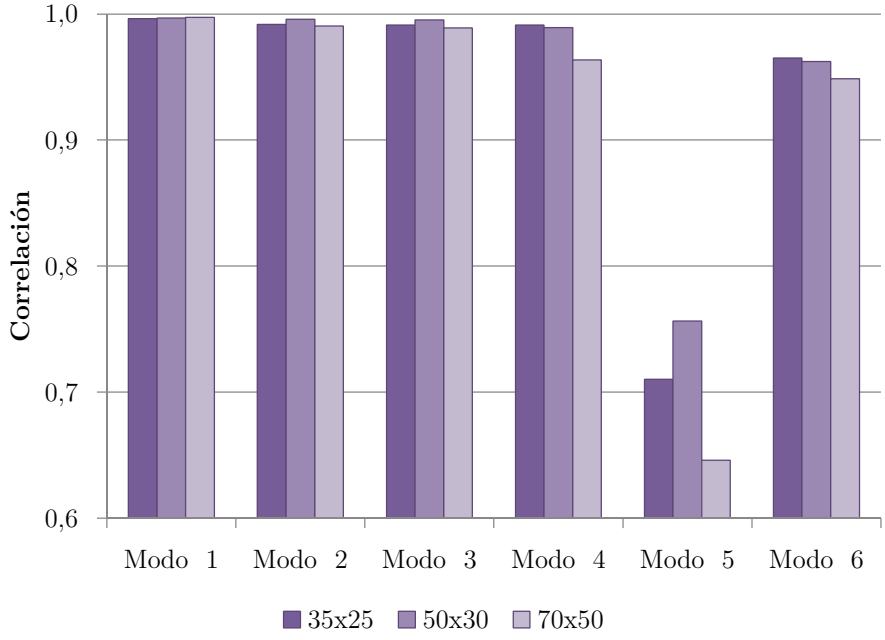


Gráfico N° 5.3: Correlación de modos experimentales y numéricos según mallado para funciones de forma de grado 12.

5.2.3.- Análisis de comportamiento frente a variación del grado polinomial

Los gráficos siguientes muestran la correlación de los seis modos estudiados para las tres mallas propuestas, agrupados para analizar la influencia del orden del polinomio.

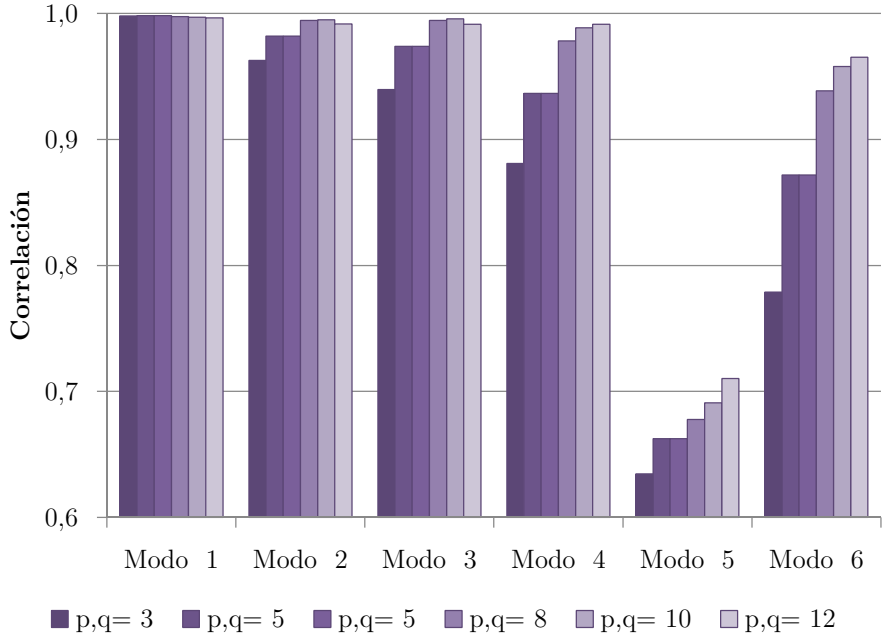


Gráfico N° 5.4: Correlación de modos según orden del polinomio para malla de 35x25 elementos.

De acuerdo a los resultados, para la malla más fina el aumento del orden polinomial no genera cambios significativos (Gráfico 5.6). Basta con observar, por ejemplo, el desempeño en el modo 5, donde el aumento total de correlación para la malla mencionada es de 0,057 mientras que para 35x25 elementos (Gráfico 5.4) se obtiene una mejora de 0,075 aun cuando en el primer caso se prueba con polinomios hasta orden 19 y el segundo hasta orden 12. Lo anterior implica que, para 70x50 elementos, aun con un mayor polinomio no se logra un mejor desempeño.

En el contexto anterior, la malla con mejores resultados en cuanto a influencia del orden del polinomio es aquella definida con 50x30 elementos, donde el aumento de correlación máximo es de 0,162 en el quinto modo (Gráfico 5.5). Esto es significativamente mayor que los otros casos estudiados.

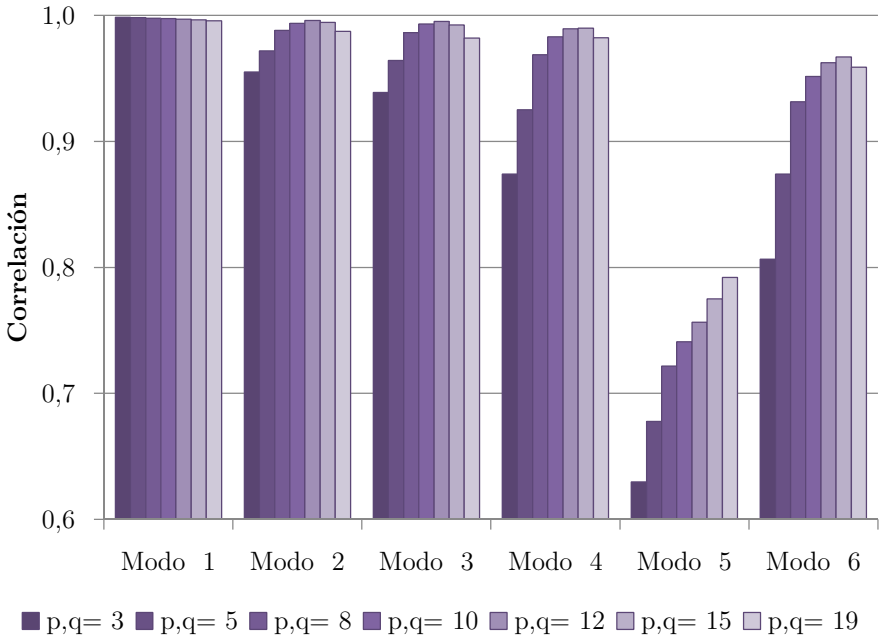


Gráfico N° 5.5: Correlación de modos según orden del polinomio para malla de 50x30 elementos.

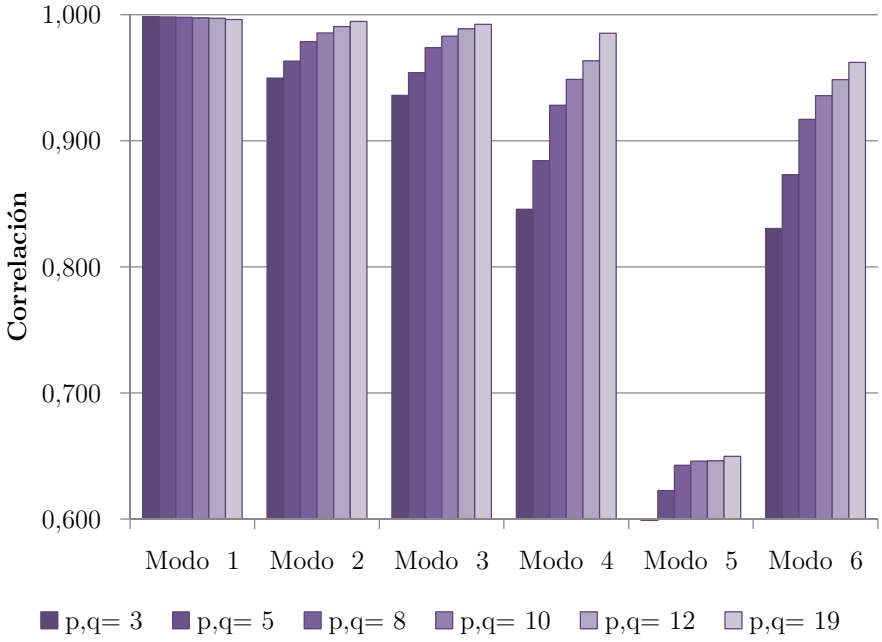


Gráfico N° 5.6: Correlación de modos según orden del polinomio para malla de 70x50 elementos.

5.2.4.- Elección de modelo más preciso

En base a los resultados de las secciones anteriores, se escoge para cada tamaño de malla el orden polinomial que presenta mejores resultados y se comparan entre mallados. Las correlaciones entre los casos mencionados se presentan en el Gráfico 5.7.

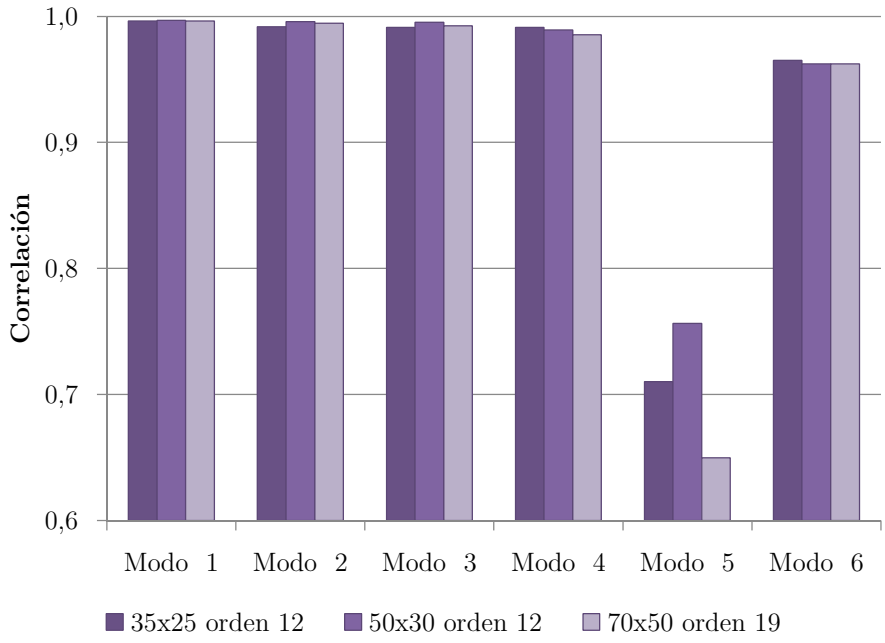


Gráfico N° 5.7: Comparación de correlación entre mejores modelos de cada mallado.

Nótese que, pese a que el gráfico 5.5 indica que la mejor correlación para el modo crítico (modo N°5) en la malla de 50x30 se obtiene para polinomio de grado 19, al considerar los errores en el cálculo de las frecuencias (Tabla 5.7) se privilegia aquel mallado que presente la mejor aproximación de ambos parámetros.

Finalmente, del Gráfico 5.7 indica que el modelo que se ajusta mejor en la mayoría de los modos es el mallado de 50x30 elementos con polinomios de orden 12.

5.3.- Modelo final

5.3.1.- Correlación numérico-experimental.

A continuación, se presentan las gráficas de los modos experimentales y numéricos obtenidos con el modelo de orden 12 con un mallado de 50x30 elementos.

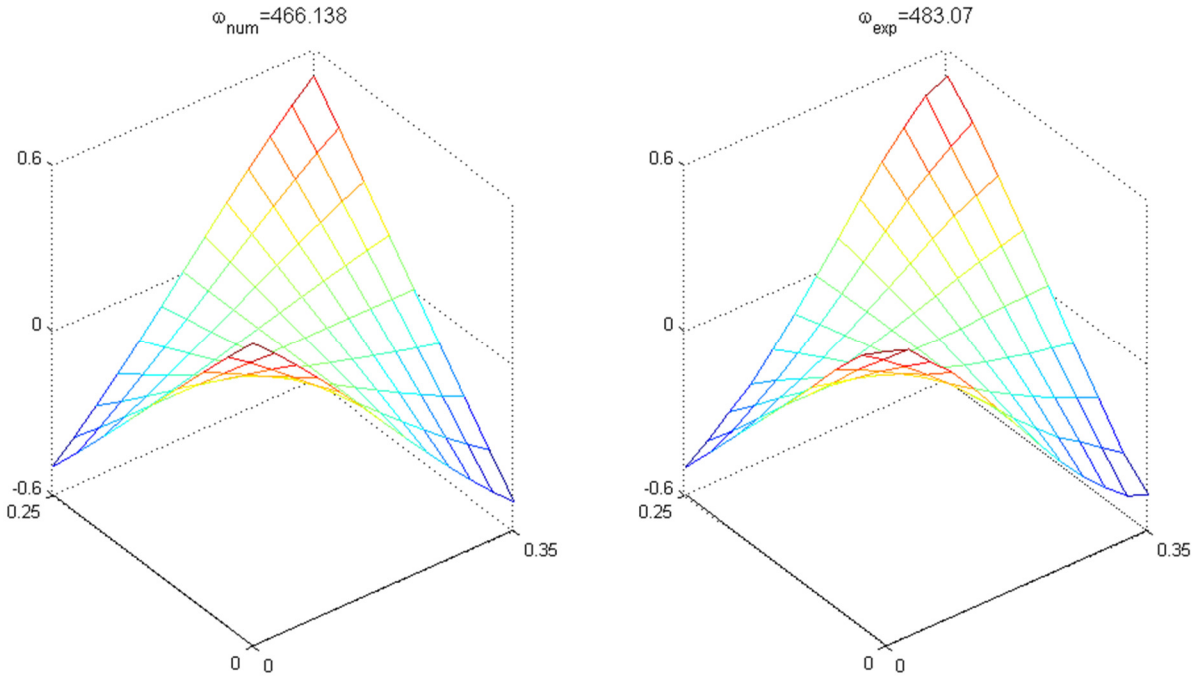


Figura N° 5.1: Modo N°1 obtenidos de manera numérica y experimental.

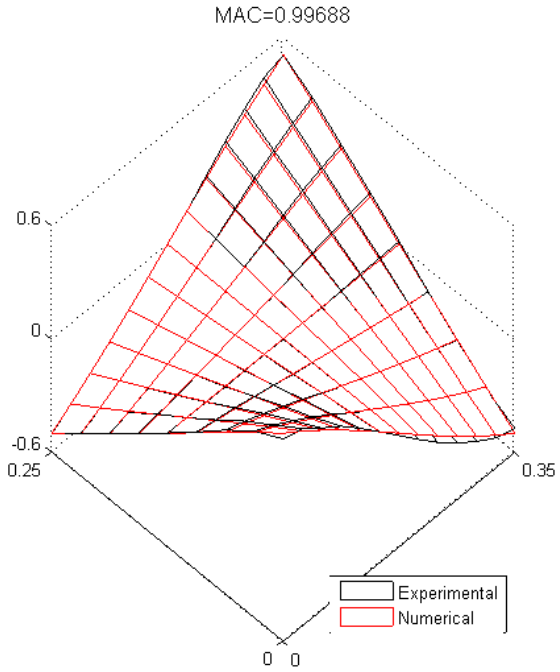


Figura N° 5.2: Correlación de modo N°1.

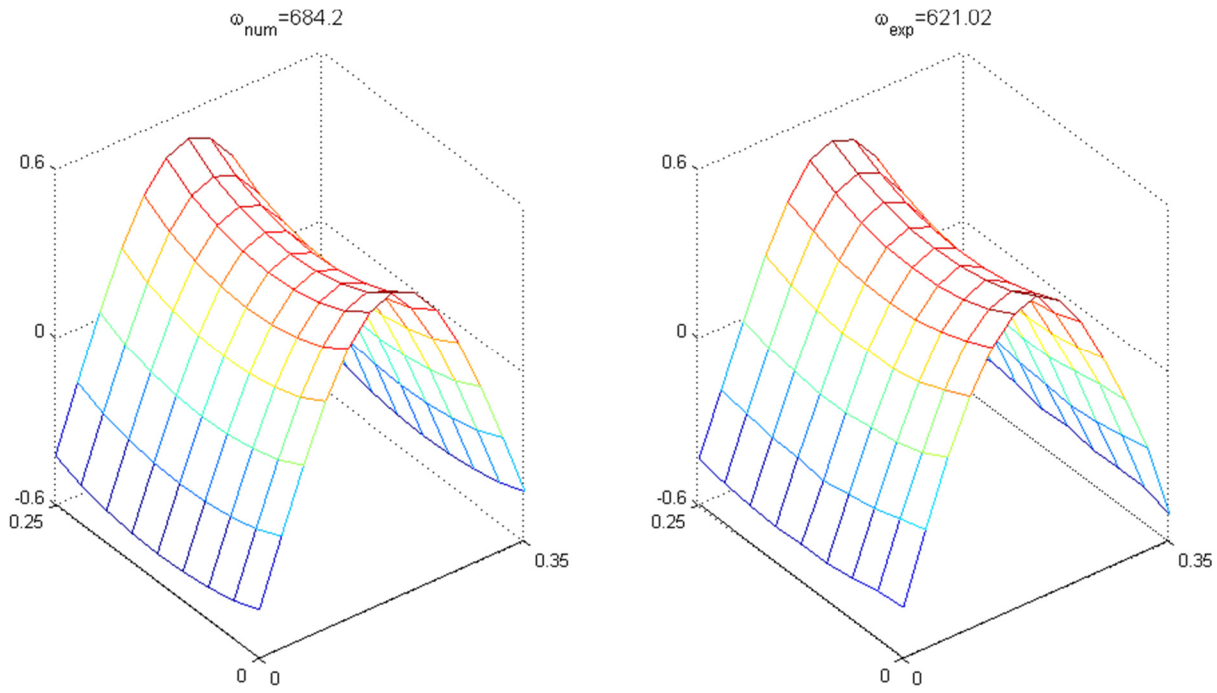


Figura N° 5.3: Modo N°2 obtenidos de manera numérica y experimental.

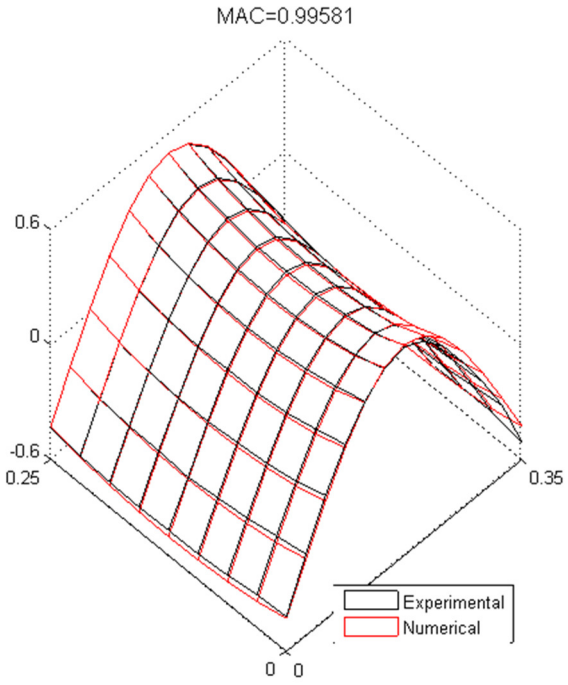


Figura N° 5.4: Correlación de modo N°2.

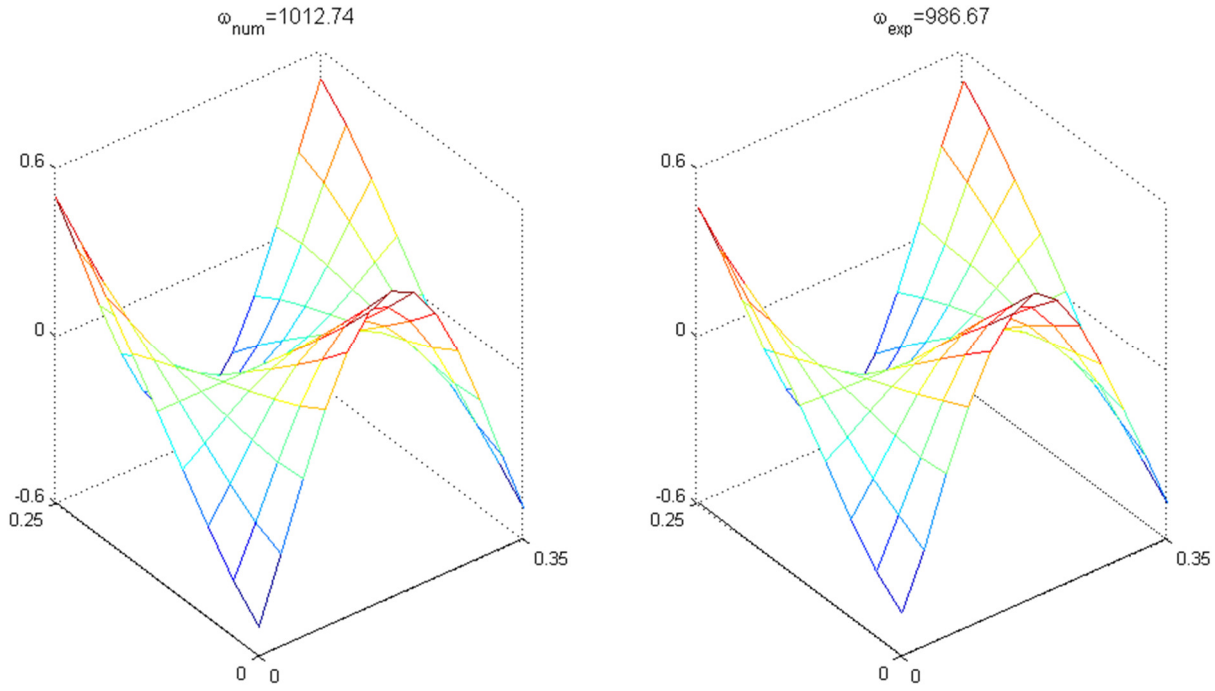


Figura N° 5.5: Modo N°3 obtenidos de manera numérica y experimental.

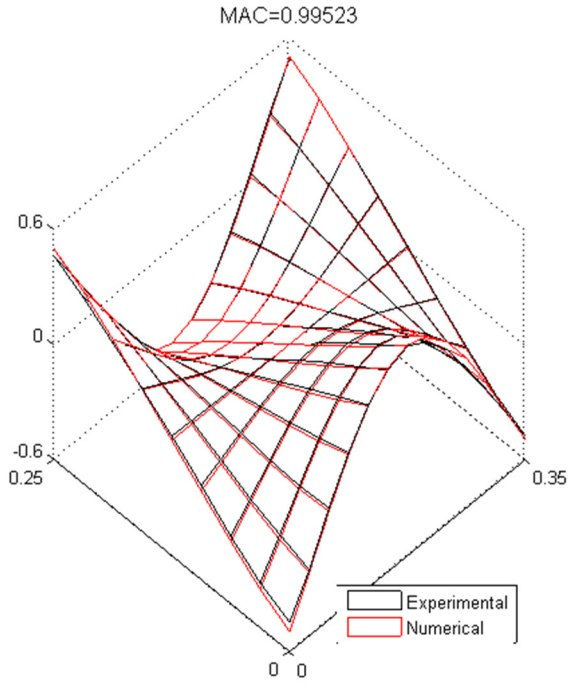


Figura N° 5.6: Correlación de modo N°3.

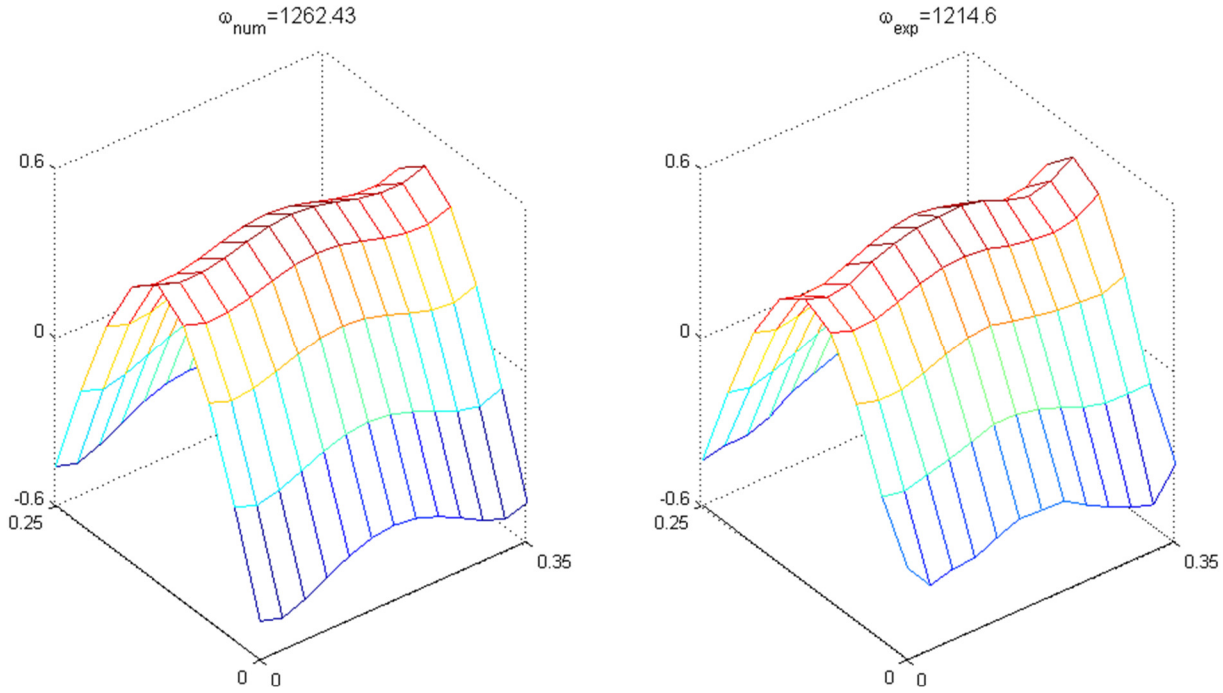


Figura N° 5.7: Modo N°4 obtenidos de manera numérica y experimental.

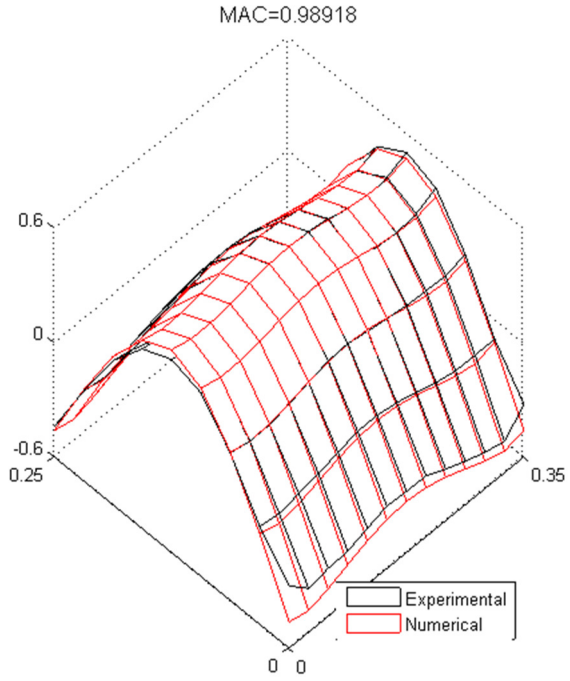


Figura N° 5.8: Correlación de modo N°4.

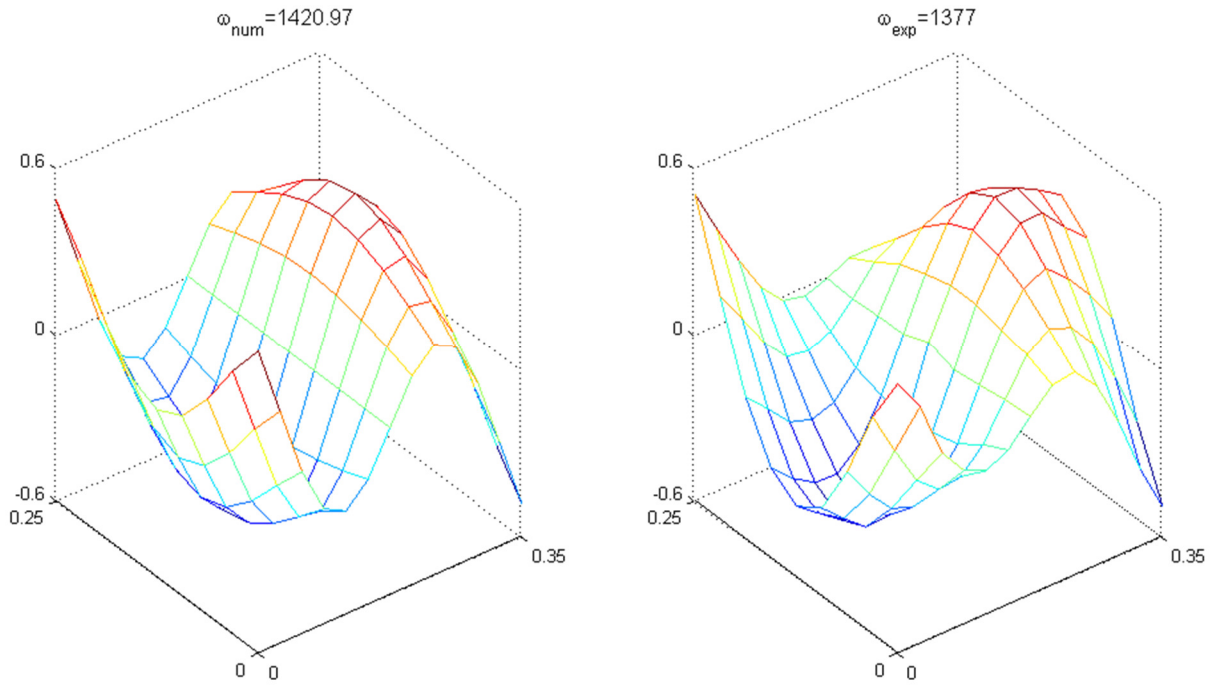


Figura N° 5.9: Modo N°5 obtenidos de manera numérica y experimental.

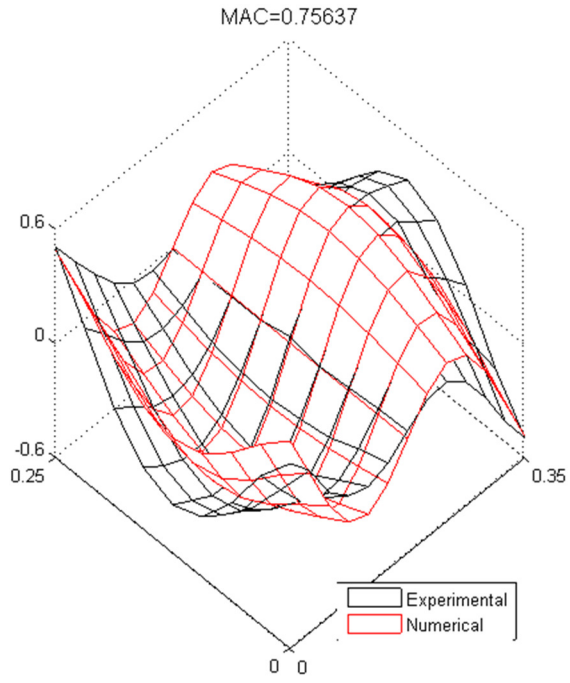


Figura N° 5.10: Correlación de modo N°5.

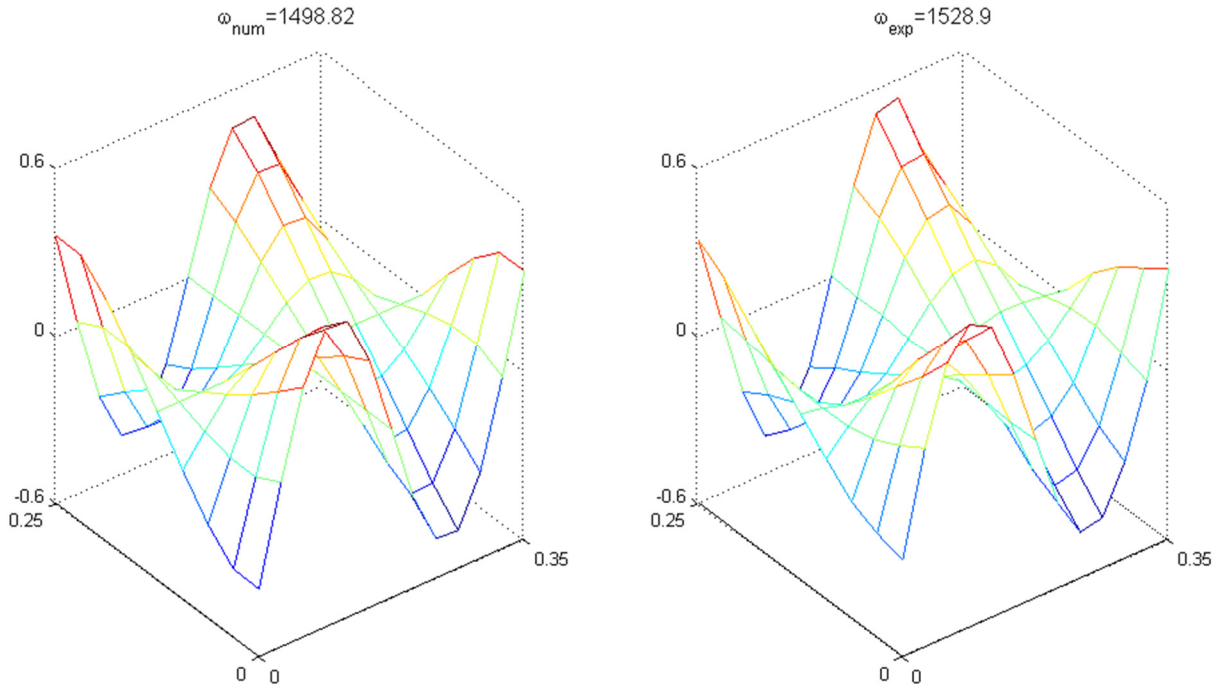


Figura N° 5.11: Modo N°6 obtenidos de manera numérica y experimental.

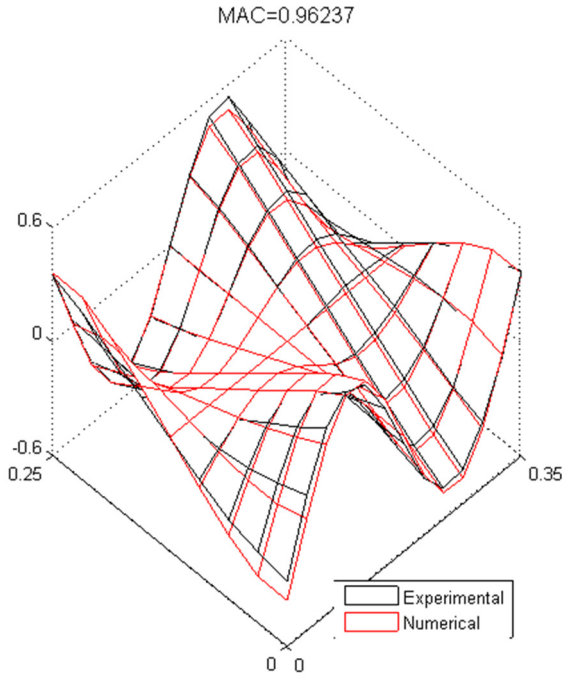


Figura N° 5.12: Correlación de modo N°6.

5.3.2.- Comparación frente a modelo FEM anterior [2].

Como se menciona en la sección 5.2.1, durante el cálculo de modos de vibración, aparece un resultado intermedio entre los modos N°4 y N°6 (numéricos), el cual no se ha caracterizado de manera experimental. En el contexto anterior, el modo que numéricamente corresponde a la sexta frecuencia, experimentalmente corresponde a la quinta y, asimismo, el séptimo modo numérico se corresponde con el sexto modo experimental.

De lo anterior, y para analizar la presencia de este "modo falso" se estudia si este resultado se repite en el modelo estudiado previamente para esta estructura con FEM [2], encontrándose que en el modelo recién mencionado, también ocurre dicho fenómeno.

Luego, resulta interesante comparar los resultados del modelo IGA con el modelo FEM. En la Tabla 5.12 se muestran comparativamente los resultados de ambos modelos.

Tabla N° 5.12: Comparación de resultados de modelo FEM e IGA.

	Frecuencia Experimental [Hz]	**Frecuencia IGA [Hz]**	**Frecuencia FEM [Hz]**	**Correlación IGA**	**Correlación FEM**
Modo 1	483,07	466,14	520,93	0,997	0,998
Modo 2	621,02	684,20	600,78	0,996	0,994
Modo 3	986,67	1.012,74	1.089,49	0,995	0,995
Modo 4	1.214,6	1.262,43	1.105,10	0,989	0,984
Modo 5	1.377	1.420,97	1.406,81	0,756	0,831
Modo 6	1.528,9	1.498,82	1.598,46	0,962	0,967

A continuación, se presentan las gráficas de las correlaciones entre ambos modelos para los primeros siete modos de vibración, considerando lo que se ha denominado como "modo falso".

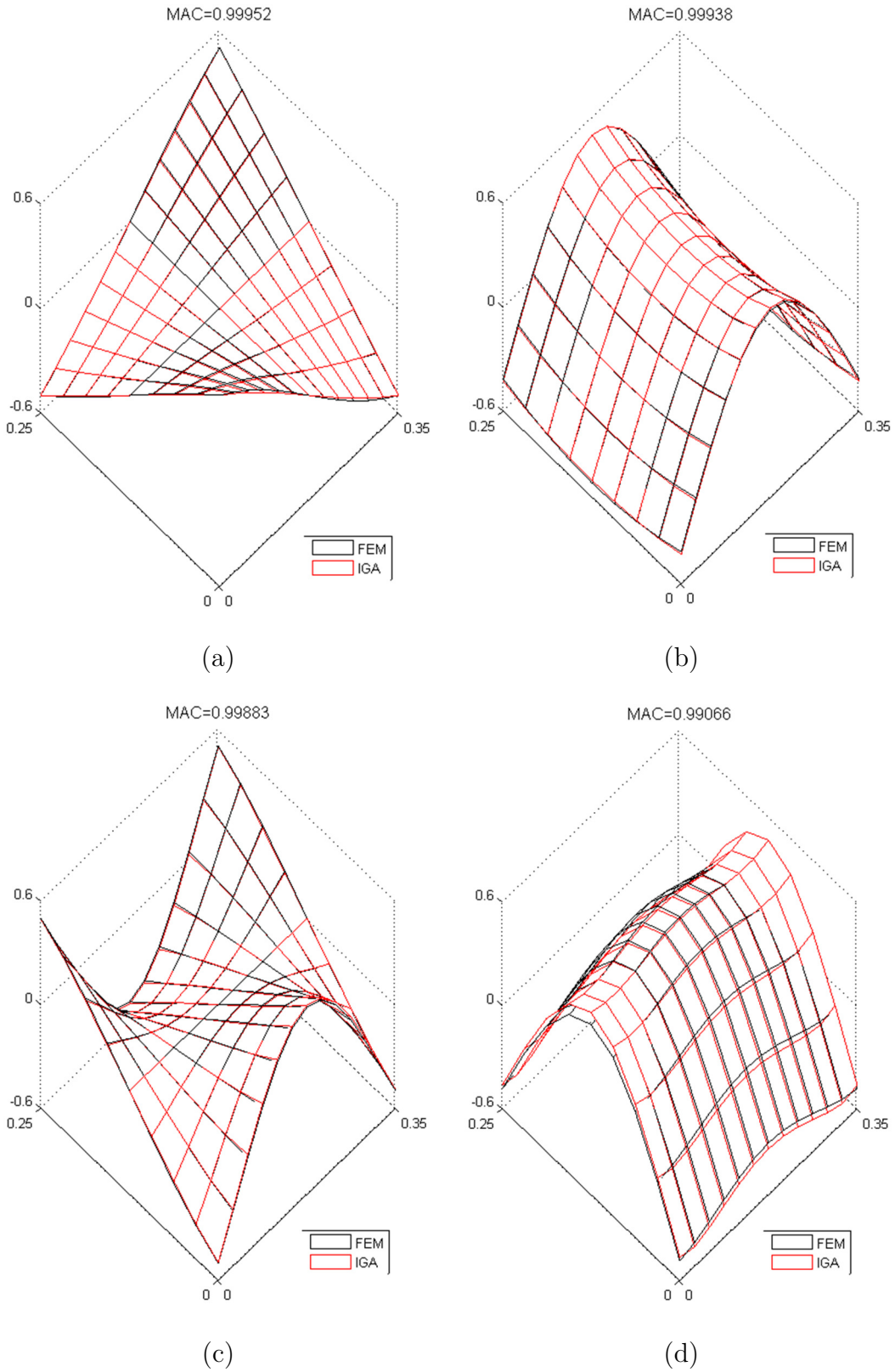


Figura N° 5.13: Correlación de modos IGA y FEM. (a) Modo 1. (b) Modo 2. (c) Modo 3. (d) Modo 4.

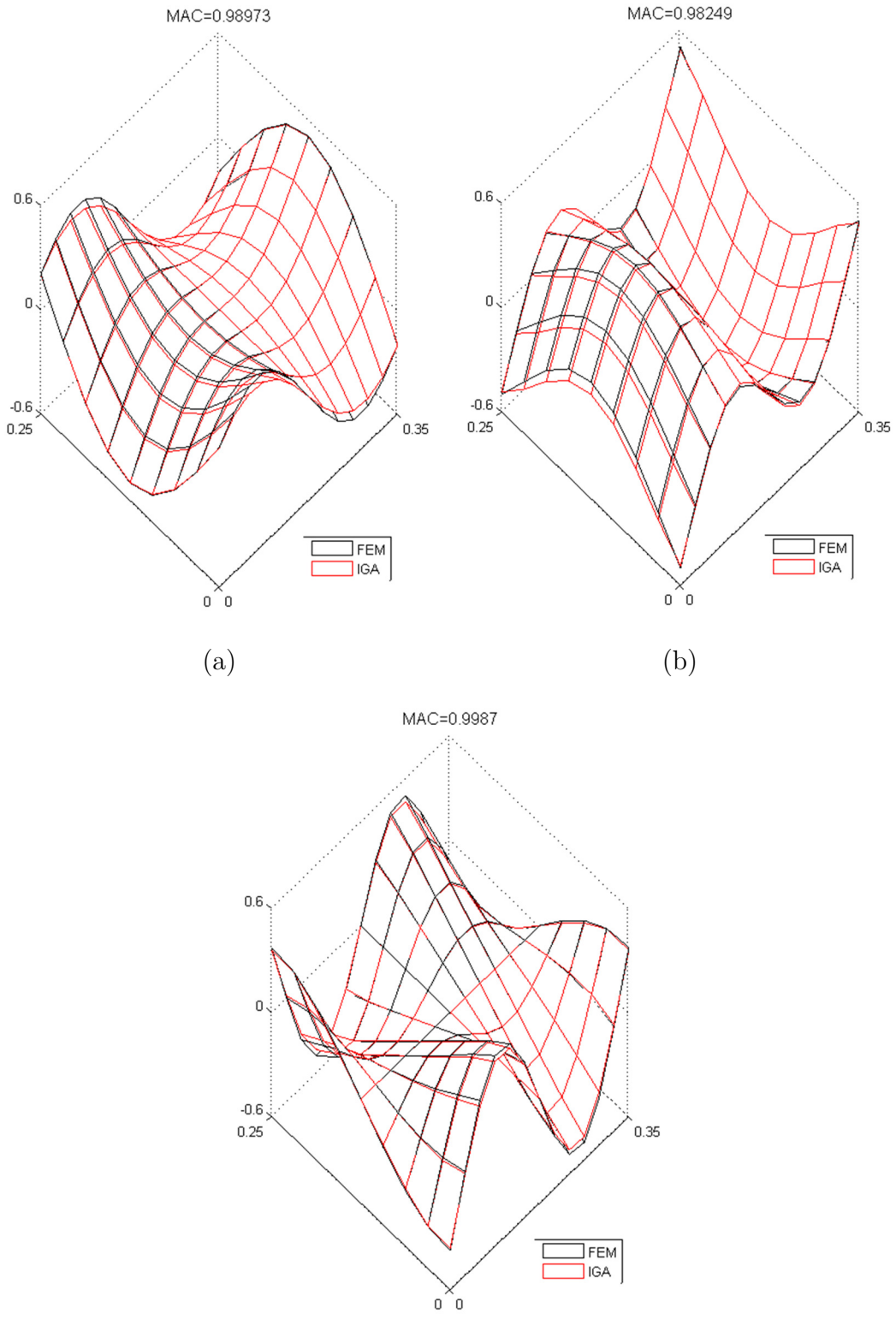


Figura N° 5.14: Correlación de modos IGA y FEM. (a) Modo 5. (b) Modo 6. (c) Modo 7.

5.3.3.- Análisis específico de modo 5

A partir de los visto en las secciones anteriores, se propone que la presencia del modo "falso" se relaciona con una medida acoplada de los modos. Para determinar si es factible que lo que experimentalmente los modos 5 y 6 (correspondientes a 5 "falso" y 5 numéricos) se hayan acoplado dando como resultado un modo mezclado, se propone comparar ese resultado con el promedio geométrico del modo numérico "falso" y el modo numérico N°5.

En la Tabla 5.13 se presentan las frecuencias numéricas, mientras que en la Tabla 5.14 se tienen los resultados del procedimiento realizado.

Tabla N° 5.13: Frecuencias naturales y error relativo porcentual para datos numéricos.

	Frecuencias numéricas [Hz]	**error relativo**
Modo falso	1317,63	---
Modo 5	1420,97	4,3%

Tabla N° 5.14: Resultado de promedio de modo 5 "falso" y modo 5 asignado.

	Frecuencia numéricas [Hz]	**error relativo**	**Correlación**
Modo 5 (promedio)	1369,30	0,6 %	0,939

Como puede verse, el procedimiento resulta en un aumento de la correlación desde 0,756 (ver Tabla 5.8) a 0,939. Además, al promediar las frecuencias el error es mínimo.

En base a lo anterior, se decide conservar el modo promedio como resultado final. Para la siguiente sección lo que se refiere a modo N°5 corresponde al modo promedio recién obtenido.

5.3.4.- Gráfico de modos de panel compuesto.

En esta sección se presentan los gráficos de modos de vibración del modelo de panel compuesto recién estudiado.

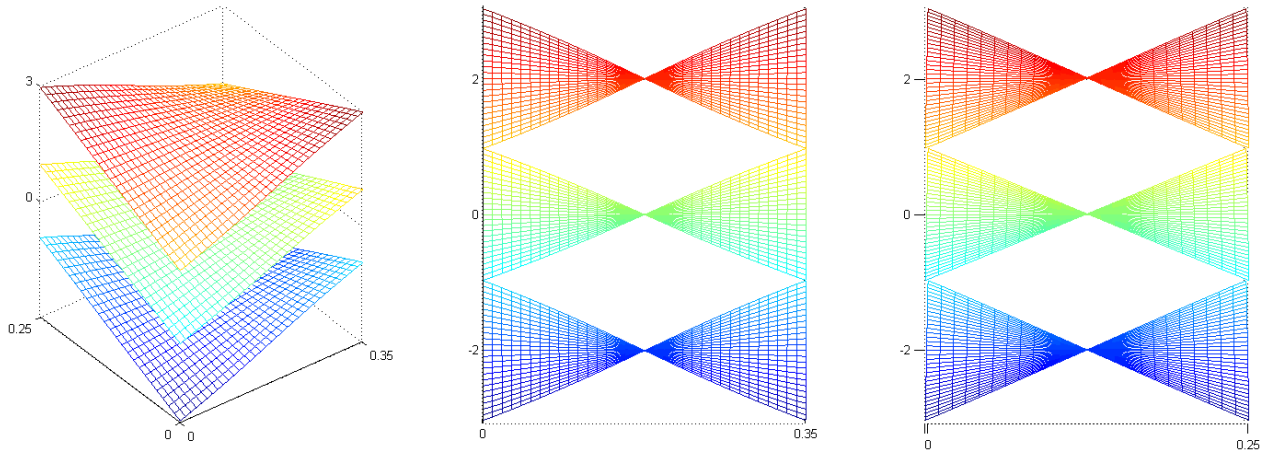


Figura N° 5.15: Modo N°1 de panel compuesto.

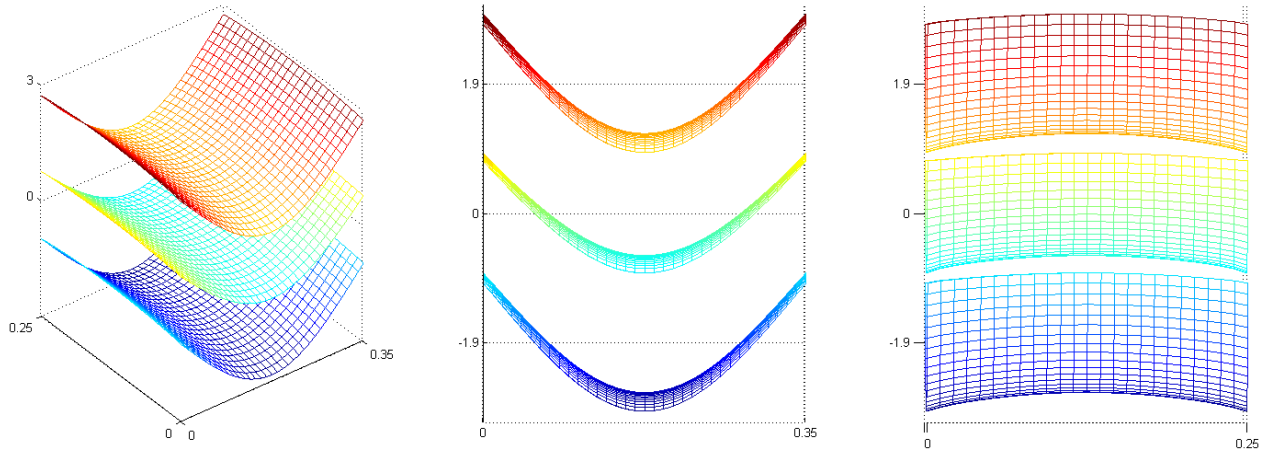


Figura N° 5.16: Modo N°2 de panel compuesto.

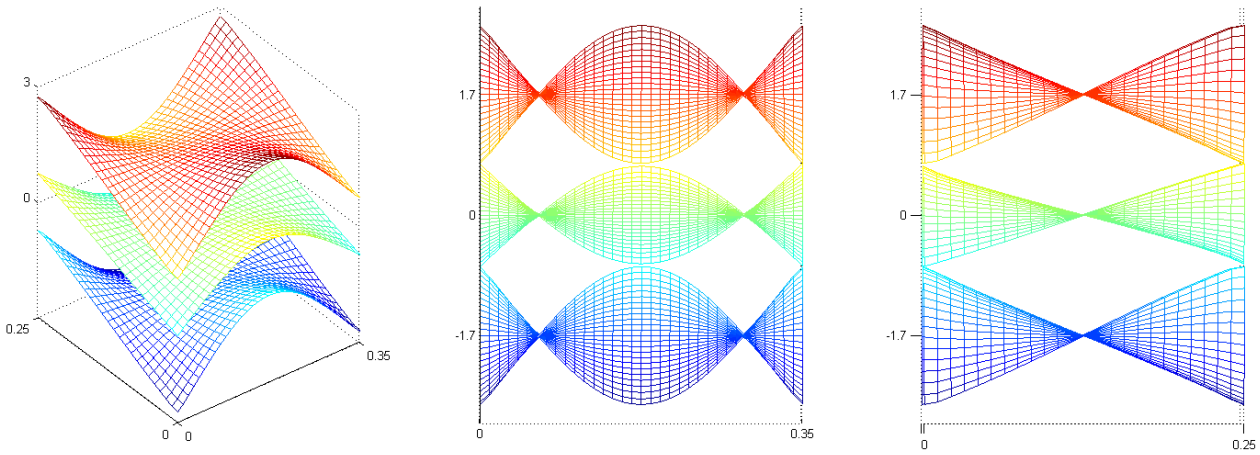


Figura N° 5.17: Modo N°3 de panel compuesto.

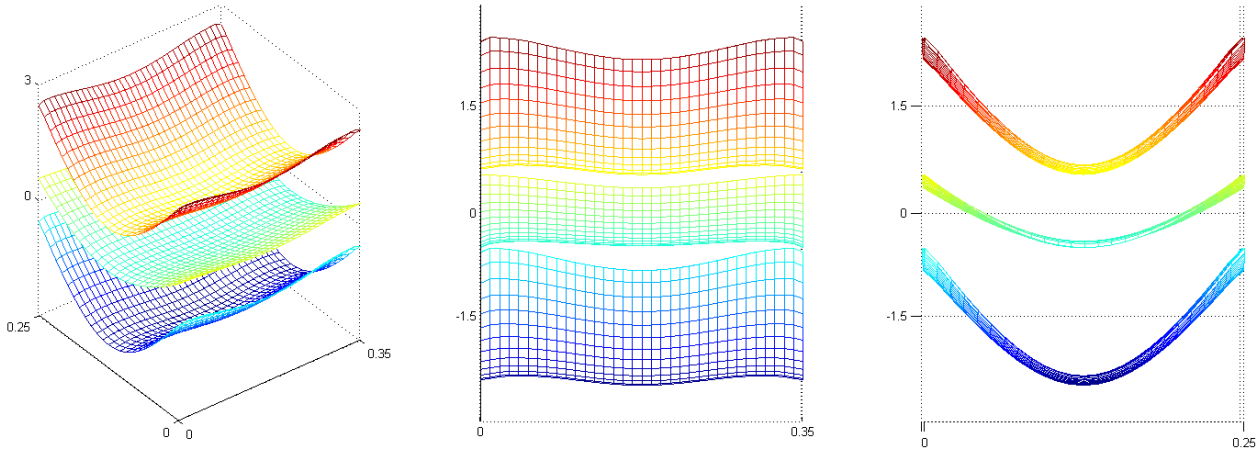


Figura N° 5.18: Modo N°4 de panel compuesto.

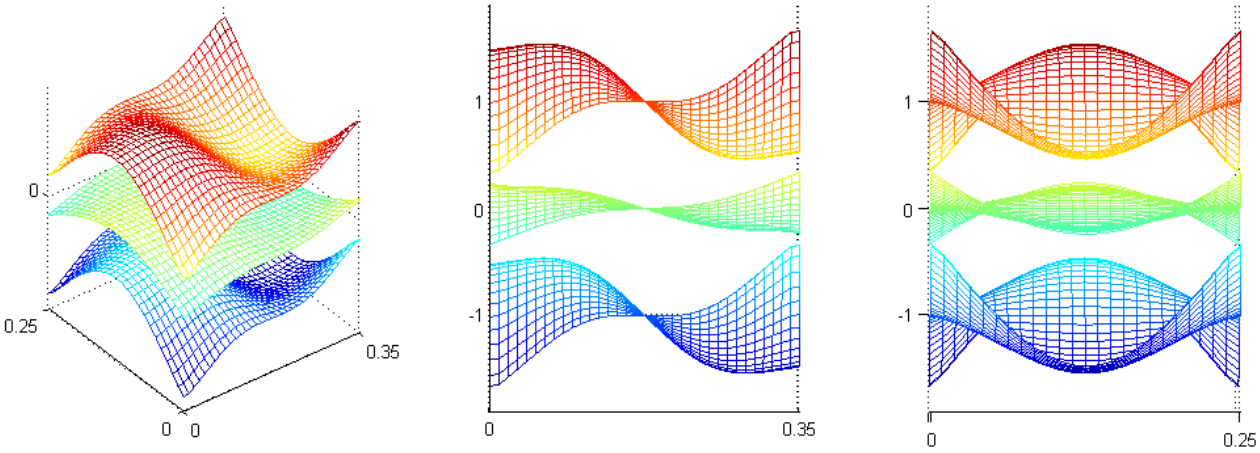


Figura N° 5.19: Modo N°5 de panel compuesto.

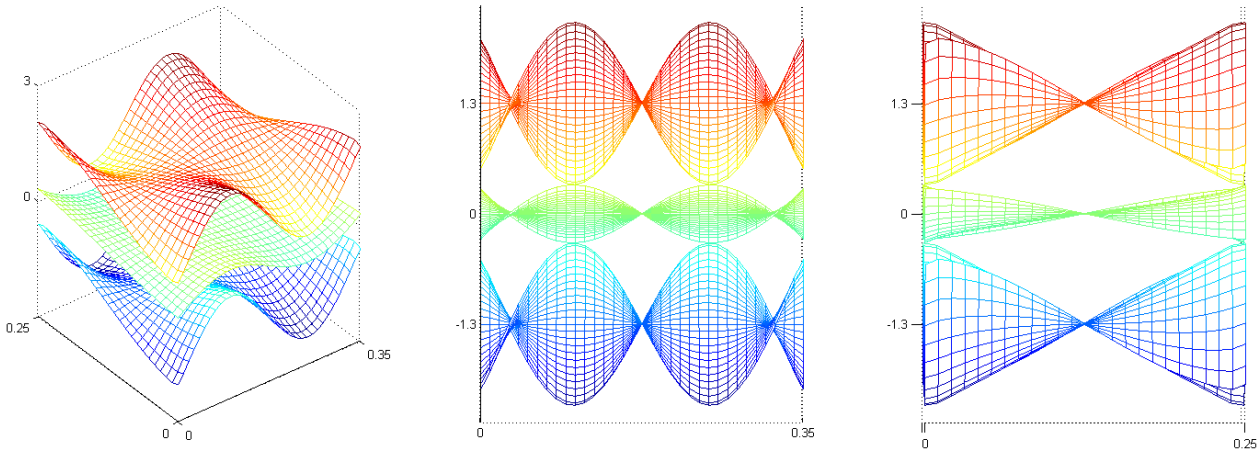


Figura N° 5.20: Modo N°6 de panel compuesto.

6.- ANÁLISIS DE RESULTADOS

Se plantea la implementación de un modelo unidimensional para el estudio de las cargas críticas de pandeo de una columna esbelta con condiciones de borde de doble empotrado, el cual se desarrolla primeramente con análisis isogeométrico y funciones de forma B-Splines y, a continuación, NURBS.

En la implementación del programa, se observa que algunas características del modelo son más sencillas de definir que en método FEM tradicional. Por ejemplo, para la variación del orden del polinomio de las funciones de forma, en el caso de IGA solo se requiere del cambio de una variable dentro del código computacional, mientras que para FEM esto conlleva una redefinición completa de las funciones de forma. Además, la definición del método IGA permite una mayor facilidad para generar cambios de malla, esto pues para aumentar o reducir el número de elementos, basta con la inserción o retiro de un punto en el vector de puntos. Lo anterior, deriva en una mayor facilidad para aumentar la precisión del modelo cuando éste se realiza mediante IGA que mediante FEM.

De acuerdo a lo estudiado, la utilización de IGA permite la obtención de resultados más exactos en este tipo de problemas, siendo su principal distinción la facilidad de variación de los parámetros de mallado (orden del polinomio y número de elementos). Lo anterior permite el cálculo de cargas críticas de pandeo más exactas, mediante la búsqueda del modelo que mejor se ajuste al problema planteado. Dicha búsqueda se realiza de manera iterativa con procedimientos simples, lo que resulta en ahorro de tiempo y trabajo.

Para la implementación del modelo bidimensional y dada la definición de las funciones utilizadas para análisis isogeométrico, se debe indagar en la teoría de métodos sin malla. Esto se relaciona con la parametrización de las funciones de forma, pues tanto las B-Splines como NURBS se encuentran definidas en toda la superficie y no solo dentro de un elemento, como en el caso de FEM. Lo anterior, resulta en una aproximación de los elementos tipo *shell* que no considera las rotaciones internas.

Al comparar los resultados del modelo IGA para una placa delgada con un análogo en FEM (Mechanical APDL) se encuentran diferencias en los cálculos de las frecuencias naturales, las que se acrecientan en la medida que se aumenta el grado polinomial. No obstante, esto pudiere no significar que dicho método es menos preciso, pues ante la ausencia de datos analíticos o experimentales, resulta difícil identificar cuál de los dos modelos (FEM o IGA) es menos exacto.

Pese a lo anterior, al observar las Figuras 4.10 a 4.21 puede verse que los modos asociados a ambos estudios son comparables, presentando los mismos tipos de deformaciones.

En el cálculo de los modos de vibración del panel compuesto tipo panal de abeja, el aumento del número de elementos no parece tener relación directa con la exactitud del modelo, es decir, mientras más grande la malla no necesariamente es más alta la correlación de los modos. En particular, se observa que cuando la malla contiene 70x50 elementos hay una disminución dicha correlación.

Respecto a lo anterior, se encuentra que la injerencia del tamaño de malla es mayor cuando el grado del polinomio es más alto (ver Gráficos 5.1, 5.2 y 5.3), lo que no significa que sea positivamente influyente.

Con relación a lo dicho, pudiere ser que el factor determinante en la influencia del tamaño de malla sea la relación de aspecto de los elementos, pues la placa es rectangular y tanto en 35x25 como en 70x50 se consideran elementos cuadrados. Luego, a pesar de que la malla de 50x30 presenta elementos rectangulares con ancho y alto traspuesto (respecto de la placa) pareciera ser que se adapta mejor a la estructura.

Por otro lado, se observa que el aumento del orden del polinomio no pareciera ser independiente del tamaño de la malla. Lo anterior puede verse en el Gráfico 5.6, donde la malla de mayor tamaño presenta una influencia mínima en el aumento de la correlación frente al incremento del orden polinomial.

Además, la tendencia al alza en la correlación sí se condice con el aumento del número de elementos para mallados más pequeños (ver los Gráficos 5.4 y 5.5). Esto puede ser un indicador de la existencia de una malla óptima, la cual permite obtener los resultados más precisos.

En vista de lo anterior, se plantea el estudio de la malla del tamaño que mejor se comporta respecto de los dos parámetros antes estudiados y se postula un modelo con 50x30 elementos. No obstante, para dicho mallado resulta interesante analizar en mayor profundidad el comportamiento del aumento del grado polinomial. En este sentido, se puede ver que para órdenes superiores a 10, el aumento de la correlación disminuye y comienza a converger a un máximo (ver Gráfico 5.5) para luego disminuir en la mayoría de los modos. Esto puede relacionarse con las funciones de forma asociadas, pues cuando el polinomio tiene un grado demasiado alto se generan más funciones de forma (aumento de un grado inserta un punto de control y por ende una función de forma adicional) las que pudiesen entorpecer el cálculo (exceso de puntos).

Lo dicho se suma al efecto en el error de cálculo de las frecuencias que genera el aumento del grado del polinomio (ver Tabla 5.7). Se plantea que el mejor ajuste en dicho parámetro se obtiene para la malla de 50x30 con 12 elementos, la cual presenta en correlación resultados muy similares al máximo (orden 15) por la convergencia estudiada anteriormente.

Para los modos obtenidos mediante el mallado propuesto (orden 12 y 50x30 elementos), se tienen en general correlaciones similares al modelo FEM que motiva esta memoria. No obstante, en el quinto modo, que es precisamente aquel que se pretende mejorar, no se logra una correlación mayor a la antes obtenida. Esto puede relacionarse con la aproximación que plantea el método IGA, que no considera ciertos grados de libertad relevantes para las vibraciones de placa.

Además, se encuentra la presencia de un modo que experimentalmente no ha sido identificado. Dicho modo corresponde correlativamente a la quinta frecuencia natural del sistema y para efectos comparativos experimentales, se ha eliminado de la serie de datos. A este modo se le ha denominado "modo falso" y también se ha presentado en el modelo FEM.

En el análisis de ambos métodos, las correlaciones experimentales se comportan de manera similar, no obstante para el caso de IGA en el modo N°5 existe un resultado considerablemente menor que el arrojado por el método FEM. En cuanto a las frecuencias, a excepción de los modos N°2 y N°5, el método isogeométrico parece aproximar de mejor manera el comportamiento del panel compuesto (ver Tabla 5.12).

Si se observan las Figuras 5.13 y 5.14, puede verse que las correlaciones de modos entre FEM e IGA son significativas, lo que pareciera indicar que ambos modelos son muy similares.

Llama la atención en particular la correlación del modo N°5 (0,98), que se constituye como el modo crítico en ambos casos, que parece ser sumamente similar entre modelos. Se puede ver que gráficamente la diferencia radica en los bordes del alto de la placa (dimensión v en el código) donde se genera una curvatura que en el caso de IGA es menos pronunciada que en el caso FEM. Si se observan las Figuras 5.9 y 5.10, puede comprobarse que el modo experimental tiene efectivamente una curvatura más pronunciada y es ahí donde pareciera radicar la diferencia del resultado entre los métodos estudiados, que favorece a FEM sobre IGA.

En la búsqueda de justificación para la presencia del modo "falso", se plantea la posibilidad de que las mediciones experimentales hayan entregado como resultado un modo acoplado, el cual debiera ser comparable con la media geométrica del modo "falso" y el modo 5 asignado de los datos numéricos. En este contexto, se encuentra que la correlación para el modo

promedio alcanza un valor de 0,939, lo que es significativamente superior a lo antes calculado.

No obstante la alta correlación, un factor determinante para identificar el acoplamiento es la similitud de frecuencias, que en este caso no apunta en dicha dirección pues existe una diferencia de aproximadamente 103 [Hz] entre el modo falso y modo 5 numérico. Sin embargo, al promediar ambos datos se obtiene un error de 0,6 % con respecto del valor experimental del modo 5. Dado que tanto la frecuencia como modos promedio se presentan como los resultados más precisos se decide adoptar dichos resultados como modo 5 (en lo que se refiere de ahora en adelante), pese a que no se encuentra una justificación razonable para lo ocurrido, puesto que los datos parecieran indicar que los modos no están acoplados.

A observar los modos de vibración de panel compuesto (Figuras 5.15 a 5.20), puede verse cómo interactúan las placas entre sí. Resulta interesante como el centro presenta claramente una mayor rigidez que las placas exteriores, lo que permite ver la influencia de dicha estructura en el comportamiento mecánico del panel compuesto.

En definitiva, se tienen los modos de vibración del panel compuesto, los que en su mayoría tienen una correlación mayor a FEM, inclusive el modo N°5 lo que se constituía como una de las motivaciones de este trabajo.

Además, si bien se sigue el planteamiento metodológico y se escoge el modelo más eficiente posible, los requerimientos computacionales del mismo son mucho mayores que los requerimientos de un análisis con FEM. Esto puede relacionarse con la forma en que se generan las matrices elementales, mientras que en FEM se tiene una única matriz elemental que se ensambla globalmente de acuerdo a los nodos y elementos definidos, IGA requiere el cómputo de cada matriz elemental, lo que deriva en varias operaciones para cada elemento. Esto último se relaciona con la definición de las NURBS, que se realiza a nivel global, exigiendo el cálculo de los valores asociados en cada punto de Gauss considerado.

Otro factor agravante es el tamaño de las matrices elementales. Mientras que para FEM con elementos *shell* de cuatro nodos las matrices son siempre de 12x12, en IGA depende del orden polinomial. Luego, sea p el orden del polinomio en x y q el orden del polinomio en y, las dimensiones de las matrices son (p+1)x(q+1), con lo cual para orden 12 éstas son de 169 nodos, lo que puede ralentizar considerablemente el proceso de ensamble.

Además, debe tenerse en cuenta que el programa de FEM contra el cual se compara corresponde a un software comercial (SD Tools para Matlab®), el cual se encuentra

optimizado para tener un desempeño rápido y efectivo, lo que no ocurre en el caso del programa IGA.

Luego, el análisis isogeomértico con funciones tipo NURBS pareciera ser computacionalmente más complejo sin la obtención de correlaciones significativamente mejores que el método FEM. No obstante, existen métodos de algoritmos de integración eficientes que pudieran hacer competitivo el tiempo de iteración de IGA contra FEM. Además, es importante mencionar que los parámetros utilizados para el modelo corresponden a valores obtenidos de la optimización del modelo FEM, por lo cual es posible mejorar el modelo IGA mediante un proceso de optimización, lo que se deja propuesto como trabajo futuro.

Finalmente, es factible que para otros problemas que tengan condiciones más adaptables al modelo IGA, por ejemplo que no requieran eliminación de grados de libertad, el resultado con este modelo sí supla los errores producidos en FEM por aproximación de geometría. Esto es incluso palpable pues, en el análisis unidimensional IGA supera holgadamente en exactitud los resultados obtenidos en FEM.

7.- CONCLUSIONES

El trabajo con un modelo IGA para el cálculo de cargas críticas y modos de pandeo en una columna esbelta doble empotrada permite la obtención de resultados de mayor exactitud que modelos FEM utilizados en el mismo problema.

El método IGA presenta mayor facilidad que FEM para la variación de parámetros de mayado (como orden del polinomio y número de elementos), lo que le confiere una ventaja al momento de encontrar el modelo óptimo con metodologías de variación de los mismos.

La implementación del modelo bidimensional de placa de Kirchhoff con IGA requiere utilización de aproximaciones sin rotación en los elementos, esto genera diferencias al momento de comparar los modos de polinomio cúbico respecto del modelo FEM. No obstante, los resultados para ambos casos son lo suficientemente similares para validar el modelo creado con IGA.

Debido a la ausencia de información experimental para placas delgadas en este trabajo, del estudio de variación de polinomio en IGA no se puede determinar cuál método (si éste o FEM) es más preciso en la aproximación de modos normales y frecuencias naturales.

En cuanto al desempeño del modelo de panel compuesto, el modo numérico N°5 no es encontrado experimentalmente, por lo cual los modos N°6 y N°7 calculados se corresponden con los modos N°5 y N°6 reales. Esto conlleva a un reordenamiento de los resultados antes de realizar los estudios de convergencia del modelo.

En relación al desempeño del modelo, se encuentra que el aumento del grado polinomial confiere más exactitud hasta aproximadamente orden 15, de ahí en más se generan mayores errores en la correlación de la mayoría de los modos en todas las mallas tratadas. Por otro lado, el aumento del tamaño de malla no es proporcional a la precisión, resultando que la malla intermedia es la que presenta mejores correlaciones.

Por otro lado, mallados con elementos rectangulares presentan un desempeño mejor que los elementos cuadrados, pues se adaptan mejor al comportamiento de la estructura.

De lo anterior, se concluye que existe una malla óptima que combina las mejores relaciones de correlación y menores errores en frecuencia, determinándose ésta en 50x30 elementos con funciones de forma de orden 12.

Respecto al modo numérico que no aparece experimentalmente (modo falso), al comparar el método IGA con FEM, se encuentra que dicho modo se presenta también en este último

caso. Esto indica que la aparición de este resultado no se constituye como un problema del método en particular, sino de la aproximación numérica del problema o bien de las mediciones experimentales.

En el contexto anterior, el modo 5 promedio resulta en una alta correlación y precisión en la frecuencia asociada. Determinándose que éste corresponde al modo obtenido de manera experimental, sin éxito en la justificación del fenómeno pues no se sustenta la hipótesis del acoplamiento de modos.

En el cálculo de modos para el mejor de los casos IGA estudiados (malla óptima), pese a que la mayoría de los modos presentan correlaciones mejores que las obtenidas con un modelo FEM previo, no se logra un aumento de la correlación del modo N°5. Lo anterior, pese a que comparativamente, los modos IGA y FEM presentan correlaciones muy similares (entre ellos). Se encuentra que la diferencia determinante es la deformación de los bordes más angostos de la placa, específicamente en la curvatura. Los errores de cálculo de IGA en este contexto se asocian con la aproximación del modelo. No obstante, se plantea un método que permite la obtención del modo 5 con mejor exactitud que el modelo FEM antes planteado.

Computacionalmente el método IGA es significativamente más demandante que FEM, lo que se relaciona con el cálculo de las matrices elementales. Esto pues para el primero se debe computar en cada elemento la matriz asociada, mientras que para el segundo se ensambla la misma matriz repetitivamente. Lo dicho se debe a la definición de las funciones de forma, que en IGA toman valores en todo el cuerpo a diferencia de FEM donde solo son válidas dentro de cada elemento. Sumado a lo anterior, las matrices elementales de IGA son de mayor tamaño que las de FEM, lo que aumenta aún más el tiempo de cálculo. Por otro lado, el modelo FEM utilizado se encuentra optimizado, por lo que los tiempos de iteración deben ser menores que el programa IGA.

Se concluye que pese a que el análisis isogeométrico no permite correlaciones significativamente mejores y computacionalmente más exigente en el análisis modal, esto puede mejorarse con algoritmos eficientes de integración y una optimización de los parámetros del modelo.

Por otro lado, el método sí constituye una alternativa mejor frente a otros problemas, como pudo comprobarse en el pandeo de una columna de Euler.

8.- BIBLIOGRAFÍA

[1] VINSON, Jack R. Sandwich structures: past, present, and future. En: Sandwich Structures 7: Advancing with Sandwich Structures and Materials. Springer Países Bajos, 2005. p. 3-12.

[2] MERUANE, V., DEL FIERRO, V., ORTIZ-BERNARDIN, A. 2014. A maximum entropy approach to assess debonding in honeycomb aluminum plates. *Entropy* 16(5): 2869-2889.

[3] NGUYEN, V.- P., BORDAS, S., RABCZUK, T. 2013. Isogeometric analysis: An overview and computer implementation aspects.

[4] REALI∗, A. 2006. An Isogeometric analysis approach for the study of structural vibrations. Journal of Earthquake Engineering,10(S1): 1-30.

[5] KIENDL, J., BLETZINGER, K.-U.,LINHARD,J, & WÜCHNER, R. 2009. Isogeometric shell analysis with Kirchhoff–Love elements. Computer Methods in Applied Mechanics and Engineering, 198(49): 3902-3914.

[6] HUGHES, T. J., COTTRELL J. A., BAZILEVS Y. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering, 194(39): 4135-4195.

[7] MERUANE V. Dinámica Estructural, Apuntes para el curso ME706 [en línea] Santiago, Chile.Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. <http://viviana.meruane.com/me706.pdf > [consulta: 12 junio 2014]

[8] PIEGL L., TILLER W. 2005. B-Spline Basis Functions. En: The NURBS book. 2ª ed. Alemania, Springer Berlin Heidelberg. pp.48-79.

[9] PIEGL L., TILLER W. 1995. Rational B-Spline Curves and Surfaces. En: The NURBS book. 2ª ed. Alemania, Springer Berlin Heidelberg. pp.117-139.

[10] LIU G. R. (2010). Meshfree Methods for Plates. En: Meshfree Methods: Moving beyond the Finite Element Method. 2ª ed. Boca Ratón, Estados Unidos de América, CRC Press. pp.475-562.

[11] SHOJAEE, S., IZADPANAH, E., VALIZADEH, N., & KIENDL, J. (2012). Free vibration analysis of thin plates by using a NURBS-based isogeometric approach. Finite Elements in Analysis and Design, 61: 23-34.

Anexos

Anexo 1: Funciones básicas en Matlab

```
function [ span ] = FindSpan( n,p,u,U )
% Adapted from The NURBS book, algorithm A2.1
% Find the span of an evaluation point u given a knotVector U. The span is
% defined by the open interval [u_i,u_i+1).
%INPUT:
% n:    number of basis functions -1.
% p:    polinomial degree.
% u:    evaluation point.
% U:    knotVector.
%OUTPUT
% span: index of the current span.

% Special case, point on the upper boundary
if u==U(n+2),span=n+1; return
end
% Define limits and start binary search
low=p; high= n+1;
mid=floor((low+high)/2);

while u<U(mid+1)||u>=U(mid+2)
    if u<U(mid+1), high=mid;
    else low=mid;
    end
    mid=floor((low+high)/2);
end
span=mid;

end
```

```
function [ N ] = BasisFuns( u,p,U )
% Adapted from The NURBS book, algorithm A2.2
% Compute the p+1 basis functions assosiated to the ith span where evaluation
% point u belongs, evaluated at u point.
%INPUT:
% u:    Evaluation point.
% p:    Polinomial degree.
% U:    knotVector.
%OUTPUT:
% N:    Vector of size 1x(p+1) containing the N_i+p to N_i basis functions
%       evaluated at point u.

N=zeros(1,p+1);
N(1)=1;
left=zeros(1,p+1);
right=left;
i=FindSpan(length(U)-p-2,p,u,U);

for j=1:p
    left(j+1)=u-U(i+1-j+1);
    right(j+1)=U(i+j+1)-u;
    saved=0;
    for r=0:j-1
        temp=N(r+1)/(right(r+2)+left(j-r+1));
        N(r+1)=saved+right(r+2)*temp;
        saved=left(j-r+1)*temp;
    end
    N(j+1)=saved;
end

end
```

```
unction [ ders ] = DerBasisFuns( u,p,U,nd )
% Adapted from The NURBS book, algorithm A2.3
% Compute the kth derivatives of the p+1 basis functions assosiated to the
```

```
% ith span where evaluation point u belongs, evaluated at u point.
%
%INPUT:
% u     :Evaluation point.
% p     :Polinomial degree.
% U     :knotVector.
% nd    :Derivation degree (if not defined only first derivative is
%       computed)
%OUTPUT:
% N     :Vector of size 1x(p+1) containing the N_i+p to N_i derivatives of
%       basis functions evaluated at point u.

ndu=zeros(p+1,p+1);
ndu(1,1)=1;
left=zeros(1,p+1);
right=left;
i=FindSpan(length(U)-p-2,p,u,U);

for j=1:p
    left(j+1)=u-U(i+1-j+1);
    right(j+1)=U(i+j+1)-u;
    saved=0;
    for r=0:j-1
        ndu(j+1,r+1)=right(r+2)+left(j-r+1);
        temp=ndu(r+1,j)/ndu(j+1,r+1);
        ndu(r+1,j+1)=saved+right(r+2)*temp;
        saved=left(j-r+1)*temp;
    end
    ndu(j+1,j+1)=saved;
end

%Computes only first derivative.
if nargin ==3, n=1;
%Compute kth derivatives
else n=nd;
end

ders=zeros(n+1,p+1);

%load Basis Functions
for j=0:p
    ders(1,j+1)=ndu(j+1,p+1);
end

%algorithm to compute derivatives
a=zeros(2,n+1);
%Loop over function index
for r=0:p
    s1=0;s2=1; %alternate rows in array a
    a(1,1)=1;

    %Loop to compute kth derivatives
    for k=1:n
        d=0;
        rk=r-k; pk=p-k;
        if r>=k
            a(s2+1,1)=a(s1+1,1)/ndu(pk+2,rk+1);
            d=a(s2+1,1)*ndu(rk+1,pk+1);
        end
        if rk>=-1, j1=1;
        else j1=-rk;
        end
        if r-1<=pk, j2=k-1;
        else j2=p-r;
        end
        for j=j1:j2
            a(s2+1,j+1)=(a(s1+1,j+1)-a(s1+1,j))/ndu(pk+2,rk+j+1);
            d=d+a(s2+1,j+1)*ndu(rk+j+1,pk+1);
        end
        if r<=pk
```

```
            a(s2+1,k+1)=-a(s1+1,k)/ndu(pk+2,r+1);
            d=d+a(s2+1,k+1)*ndu(r+1,pk+1);
        end
        ders(k+1,r+1)=d;
        j=s1; s1=s2; s2=j; %Switch rows
    end
end

% Multiply by correct factor
r=p;
for k=1:n
    for j=0:p
        ders(k+1,j+1)=ders(k+1,j+1)*r;
    end
    r=r*(p-k);
end



end
```

```
function [] = Line_B_Spline( L,p,insertionPoints,derdegree,res)
% Grafica funciones de Forma B-Spline y sus derivadas para una barra o
% viga, creando automáticamente un knotVector abierto.
%
%INPUT:
% L:                Largo de la viga.
% p:                Grado del polinomio
% insertionPoints:  Número de knots que se desea insertar.
% derdegree:        Grado de derivación que se desea graficar (cero para
%                   funciones de base)
% res:              # de puntos a considerar para los gráficos
%OUTPUT:            derdegree+1 gráficos.

%=============Definir los parámetros de la viga============================
knotVector=[zeros(1,p) linspace(0,L,insertionPoints+2) L*ones(1,p)]; %Se define con multiplicidad
p+1
n=length(knotVector)-p-1;
ControlPoints=linspace(0,L,n); %Se crean n puntos de control.

%================Gráficos de funciones de Base============================
%Datos para graficar
graf=linspace(knotVector(1),knotVector(end)*(1-1e-5),res); %Vector a graficar
Basis=zeros(length(graf),n,derdegree+1); %Matriz que guarda las funciones de forma
% Evaluar y guardar los valores:
for g=1:length(graf)
    i=FindSpan(n-1,p,graf(g),knotVector);
    N=DerBasisFuns(graf(g),p,knotVector,derdegree);
    for k=1:derdegree+1
        for aux=0:p
            Basis(g,i-p+aux+1,k)=N(k,aux+1);
        end
    end
end


%Plotear las funciones de base
plotVector=unique(knotVector);
scrs=get(0,'screenSize');
figure('num','off','name','Funciones de Forma','OuterPosition',[scrs(3)*0.05 scrs(4)*0.05
scrs(3)*0.9 scrs(4)*0.95])
set(gcf,'Color',[1 1 1])
hold on
for i=1:n
    Color=rand(1,3);
    plot(graf,Basis(:,i,1),'Color',Color,'linewidth',2)
    legendInfo{i}=['N_{',num2str(i),',',num2str(p),'}'];
end
plot(plotVector,zeros(1,length(plotVector)),'ko','linewidth',5)
```

```
for j=2:length(plotVector)
plot([plotVector(j) plotVector(j)],[0 1],'k--')
end
hold off
legend(legendInfo,'Location','EastOutside')
title(['Basis Functions of degree ', num2str(p),' and ',num2str(insertionPoints),' knots
inserted'],'fontsize',16,'color',[0 0 0])

if derdegree>0

    %Plot derivatives Forms
    for der=1:derdegree
        figure('num','off','OuterPosition',[scrs(3)*0.05 scrs(4)*0.05 scrs(3)*0.9 scrs(4)*0.95])
        set(gcf,'Color',[1 1 1])
        downlimit=min(min(Basis(:,:,der+1)));
        uplimit=max(max(Basis(:,:,der+1)));
        hold on
        for i=1:n
            Color=rand(1,3);
            plot(graf,Basis(:,i,der+1),'Color',Color,'linewidth',2)
            legendInfo{i}=['N_{',num2str(i),',',num2str(p),'}'];
        end
        plot(plotVector,downlimit*ones(1,length(plotVector)),'ko','linewidth',5)
        for j=2:length(plotVector)
        plot([plotVector(j) plotVector(j)],[downlimit uplimit],'k--')
        end
        hold off
        legend(legendInfo,'Location','EastOutside')
        title([num2str(der),' derivative of Basis Functions of degree ', num2str(p),' and
',num2str(insertionPoints),' knots inserted'],'fontsize',16,'color',[0 0 0])
        axis([0 L downlimit uplimit])
    end
end



end
```

```
unction [] = Line_NURBS( L,p,insertionPoints,derdegree,res)
% Grafica funciones de Forma NURBS y sus derivadas para una barra o
% viga, creando automáticamente un knotVector abierto.
%
%INPUT:
% L:                Largo de la viga.
% p:                Grado del polinomio
% insertionPoints:  Número de knots que se desea insertar.
% derdegree:        Grado de derivación que se desea graficar (al menos 1)
% res:              # de puntos a considerar para los gráficos
%OUTPUT:            derdegree+1 gráficos.

if derdegree<1
    error('Debe ingresar al menos 1 derivada')
    return
end

%=============Definir los parámetros de la viga============================
knotVector=[zeros(1,p) linspace(0,L,insertionPoints+2) L*ones(1,p)]; %Se define con multiplicidad
p+1
n=length(knotVector)-p-1;
ControlPoints=linspace(0,L,n); %Se crean n puntos de control.
Wheights=ones(1,length(ControlPoints));

%==============================Gráficos====================================
%Datos para graficar
graf=linspace(knotVector(1),knotVector(end)*(1-1e-5),res); %Vector a graficar
Basis=zeros(length(graf),n,derdegree+1); %Matriz que guarda las funciones de forma
BasisAux=zeros(length(graf),n,derdegree+1); %Guarda las funciones de forma no ponderadas.
Weigths=ones(1,n); %En estas estructuras no se requiere pesos diferentes
suma=zeros(derdegree+1,1);
```

```
% Evaluar y guardar los valores:

for g=1:length(graf)
    i=FindSpan(n-1,p,graf(g),knotVector);
    N=DerBasisFuns(graf(g),p,knotVector,derdegree);


    %Ponderar todas las derivadas por el peso correspondiente y se calculan
    %las sumas
    for i2=1:derdegree+1
    N(i2,:)=N(i2,:).*Weigths(i-p+1:i+1);
    suma(i2)=sum(N(i2,:));
    end
    %Asignar valores de funciones de forma
    BasisAux(g,i-p+1:i+1,1)=N(1,:);
    Basis(g,:,1)=BasisAux(g,:,1)/suma(1);

    for k=2:derdegree+1
        for aux=0:p
            temp=0;
            for i2=1:k
                temp=temp+factorial(k)/(factorial(i2)*factorial(k-i2))*suma(i2)*BasisAux(g,i-
p+aux+1,k-i2+1);
            end
            BasisAux(g,i-p+aux+1,k)=(N(k,aux+1)*Wheights(i-p+aux+1)-temp);
        end

    end
end
%Plotear las funciones de base
plotVector=unique(knotVector);
scrs=get(0,'screenSize');
figure('num','off','name','Funciones de Forma','OuterPosition',[scrs(3)*0.05 scrs(4)*0.05
scrs(3)*0.9 scrs(4)*0.95])
set(gcf,'Color',[1 1 1])
hold on
for i=1:n
    Color=rand(1,3);
    plot(graf,Basis(:,i,1),'Color',Color,'linewidth',2)
    legendInfo{i}=['N_{',num2str(i),',',num2str(p),'}'];
end
plot(plotVector,zeros(1,length(plotVector)),'ko','linewidth',5)
for j=2:length(plotVector)
plot([plotVector(j) plotVector(j)],[0 1],'k--')
end
hold off
legend(legendInfo,'Location','EastOutside')
title(['Basis Functions of degree ', num2str(p),' and ',num2str(insertionPoints),' knots
inserted'],'fontsize',16,'color',[0 0 0])

%Plot derivatives Forms
% for der=1:derdegree
%     figure('num','off','OuterPosition',[scrs(3)*0.05 scrs(4)*0.05 scrs(3)*0.9 scrs(4)*0.95])
%     set(gcf,'Color',[1 1 1])
%     downlimit=min(min(Basis(:,:,der+1)));
%     uplimit=max(max(Basis(:,:,der+1)));
%     hold on
%     for i=1:n
%         Color=rand(1,3);
%         plot(graf,Basis(:,i,der+1),'Color',Color,'linewidth',2)
%         legendInfo{i}=['N_{',num2str(i),',',num2str(p),'}'];
%     end
%     plot(plotVector,downlimit*ones(1,length(plotVector)),'ko','linewidth',5)
%     for j=2:length(plotVector)
%     plot([plotVector(j) plotVector(j)],[downlimit uplimit],'k--')
%     end
%     hold off
%     legend(legendInfo,'Location','EastOutside')
%     title([num2str(der),' derivative of Basis Functions of degree ', num2str(p),' and
',num2str(insertionPoints),' knots inserted'],'fontsize',16,'color',[0 0 0])
```

```
%     axis([0 L downlimit uplimit])
% end


end
```

```
function [ s w ] = GaussCuad( p,a,b )
% Entrega los parámetros para la cuadratura de Gauss en un intervalo [a,b]
% con a~-1 y b~1.
%
%INPUT:
% p:    grado del polinomio a integrar, se crean p+1 puntos.
% a:    límite inferior.
% b:    límite superior.
% OUTPUT:
% s:    puntos de la cuadratura
% w:    pesos asociados a cada punto.

if p+1>20
    error('Not valid for polinomial degree over 19')
end

h=(b-a)/2;
m=(a+b)/2;
s=gausspoints1d(p+1)*h+m*ones(1,p+1);
w=gaussweights1d(p+1);

%Funciones anexas
function x = gausspoints1d(n)
x = ones(1,n);

  if ( n == 1 )
    x(1) = 0.0;
  elseif ( n == 2 )
    x(1) = - 0.57735026918962576450914878050 2;
    x(2) =   0.57735026918962576450914878050 2;
  elseif ( n == 3 )
    x(1) = - 0.77459666924148337703585307995 6;
    x(2) =  0.000000000000000;
    x(3) =   0.77459666924148337703585307995 6;
  elseif ( n == 4 )
    x(1) = - 0.86113631159405257522394648889 3;
    x(2) = - 0.33998104358485626480266575910 3;
    x(3) =   0.33998104358485626480266575910 3;
    x(4) =   0.86113631159405257522394648889 3;
  elseif ( n == 5 )
    x(1) = - 0.90617984593866399279762687829 9;
    x(2) = - 0.53846931010568309103631442070 0;
    x(3) =   0.0;
    x(4) =   0.53846931010568309103631442070 0;
    x(5) =   0.90617984593866399279762687829 9;
  elseif ( n == 6 )
    x(1) = - 0.93246951420315202781230155449 4;
    x(2) = - 0.66120938646626451366139959502 0;
    x(3) = - 0.23861918608319690863050172168 1;
    x(4) =   0.23861918608319690863050172168 1;
    x(5) =   0.66120938646626451366139959502 0;
    x(6) =   0.93246951420315202781230155449 4;
  elseif ( n == 7 )
    x(1) = - 0.94910791234275852452618968404 8;
    x(2) = - 0.74153118559939443986386477328 1;
    x(3) = - 0.40584515137739716690660641207 7;
    x(4) =   0.0;
    x(5) =   0.40584515137739716690660641207 7;
    x(6) =   0.74153118559939443986386477328 1;
    x(7) =   0.94910791234275852452618968404 8;
  elseif ( n == 8 )
    x(1) = - 0.96028985649753623168356086856 9;
    x(2) = - 0.79666647741362673959155393647 6;
```

```
  x(3) = - 0.525532409916328985817739049189;
  x(4) = - 0.183434642495649804939476142360;
  x(5) =   0.183434642495649804939476142360;
  x(6) =   0.525532409916328985817739049189;
  x(7) =   0.796666477413626739591553936476;
  x(8) =   0.960289856497536231683560868569;
elseif ( n == 9 )
  x(1) = - 0.968160239507626089835576202904;
  x(2) = - 0.836031107326635794299429788070;
  x(3) = - 0.613371432700590397308702039341;
  x(4) = - 0.324253423403808929038538014643;
  x(5) =   0.0;
  x(6) =   0.324253423403808929038538014643;
  x(7) =   0.613371432700590397308702039341;
  x(8) =   0.836031107326635794299429788070;
  x(9) =   0.968160239507626089835576202904;
elseif ( n == 10 )
  x(1) =  - 0.973906528517171720077964012084;
  x(2) =  - 0.865063366688984510732096688423;
  x(3) =  - 0.679409568299024406234327365115;
  x(4) =  - 0.433395394129247290799265943166;
  x(5) =  - 0.148874338981631210884826001130;
  x(6) =    0.148874338981631210884826001130;
  x(7) =    0.433395394129247290799265943166;
  x(8) =    0.679409568299024406234327365115;
  x(9) =    0.865063366688984510732096688423;
  x(10) =   0.973906528517171720077964012084;
elseif (n == 11)
  x(1)=-0.978228658146056992803938001123;
  x(2)=-0.887062599768095299075157769304;
  x(3)=-0.730152005574049324093416252031;
  x(4)=-0.519096129206811815925725669459;
  x(5)=-0.269543155952344972331531985401;
  x(6)= 0;
  x(7)= 0.269543155952344972331531985401;
  x(8)= 0.519096129206811815925725669459;
  x(9)= 0.730152005574049324093416252031;
  x(10)= 0.887062599768095299075157769304;
  x(11)= 0.978228658146056992803938001123;
elseif (n == 12)
  x(1)=-0.981560634246719250690549090149;
  x(2)=-0.904117256370474856678465866119;
  x(3)=-0.769902674194304687036893833213;
  x(4)=-0.587317954286617447296702418941;
  x(5)=-0.367831498998180193752691536644;
  x(6)=-0.125233408511468915472441369464;
  x(7)= 0.125233408511468915472441369464;
  x(8)= 0.367831498998180193752691536644;
  x(9)= 0.587317954286617447296702418941;
  x(10)= 0.769902674194304687036893833213;
  x(11)= 0.904117256370474856678465866119;
  x(12)= 0.981560634246719250690549090149;
elseif (n == 13)
  x(1)=-0.984183054718588149472829448807;
  x(2)=-0.917598399222977965206547836501;
  x(3)=-0.801578090733309912794206489583;
  x(4)=-0.642349339440340220643984606996;
  x(5)=-0.448492751036446852877912852128;
  x(6)=-0.230458315955134794065528121098;
  x(7)= 0;
  x(8)= 0.230458315955134794065528121098;
  x(9)= 0.448492751036446852877912852128;
  x(10)= 0.642349339440340220643984606996;
  x(11)= 0.801578090733309912794206489583;
  x(12)= 0.917598399222977965206547836501;
  x(13)= 0.984183054718588149472829448807;
elseif (n == 14)
  x(1)=-0.986283808696812338841597266704;
  x(2)=-0.928434883663573517336391139378;
  x(3)=-0.827201315069764993189794742650;
```

```
  x(4)=-0.68729290481168547014801980 3019;
  x(5)=-0.51524863635815409196529071 8551;
  x(6)=-0.31911236892788976043567182 4168;
  x(7)=-0.10805494870734366206624465 0220;
  x(8)= 0.10805494870734366206624465 0220;
  x(9)= 0.31911236892788976043567182 4168;
  x(10)= 0.51524863635815409196529071 8551;
  x(11)= 0.68729290481168547014801980 3019;
  x(12)= 0.82720131506976499318979474 2650;
  x(13)= 0.92843488366357351733639113 9378;
  x(14)= 0.98628380869681233884159726 6704;
elseif (n == 15)
  x(1)=-0.98799251802048542848956571 8587;
  x(2)=-0.93727339240070590430775894 7710;
  x(3)=-0.84820658341042721620064832 0774;
  x(4)=-0.72441773136017004741618605 4614;
  x(5)=-0.57097217260853884753722673 7254;
  x(6)=-0.39415134707756336989720737 0981;
  x(7)=-0.20119409399743452230062830 3395;
  x(8)= 0;
  x(9)= 0.20119409399743452230062830 3395;
  x(10)= 0.39415134707756336989720737 0981;
  x(11)= 0.57097217260853884753722673 7254;
  x(12)= 0.72441773136017004741618605 4614;
  x(13)= 0.84820658341042721620064832 0774;
  x(14)= 0.93727339240070590430775894 7710;
  x(15)= 0.98799251802048542848956571 8587;
elseif (n == 16)
  x(1)=-0.98940093499164993259615417 3450;
  x(2)=-0.94457502307323257607798841 5535;
  x(3)=-0.86563120238783174388046789 7712;
  x(4)=-0.75540440835500303389510119 4847;
  x(5)=-0.61787624440264374844667176 4049;
  x(6)=-0.45801677765722738634241944 2984;
  x(7)=-0.28160355077925891323046050 1460;
  x(8)=-0.09501250983763744018531933 54250;
  x(9)= 0.09501250983763744018531933 54250;
  x(10)= 0.28160355077925891323046050 1460;
  x(11)= 0.45801677765722738634241944 2984;
  x(12)= 0.61787624440264374844667176 4049;
  x(13)= 0.75540440835500303389510119 4847;
  x(14)= 0.86563120238783174388046789 7712;
  x(15)= 0.94457502307323257607798841 5535;
  x(16)= 0.98940093499164993259615417 3450;
elseif (n == 17)
  x(1)=-0.99057547531441733567543401 9941;
  x(2)=-0.95067552176876776122271695 7896;
  x(3)=-0.88023915372698590212295569 4488;
  x(4)=-0.78151400389680140692523005 5520;
  x(5)=-0.65767115921669076585030221 6643;
  x(6)=-0.51269053708647696788624656 8630;
  x(7)=-0.35123176345387631529718551 7095;
  x(8)=-0.17848418149584785585067749 3654;
  x(9)= 0;
  x(10)= 0.17848418149584785585067749 3654;
  x(11)= 0.35123176345387631529718551 7095;
  x(12)= 0.51269053708647696788624656 8630;
  x(13)= 0.65767115921669076585030221 6643;
  x(14)= 0.78151400389680140692523005 5520;
  x(15)= 0.88023915372698590212295569 4488;
  x(16)= 0.95067552176876776122271695 7896;
  x(17)= 0.99057547531441733567543401 9941;
elseif (n == 18)
  x(1)=-0.99156516842093094673001600 4706;
  x(2)=-0.95582394957139775518119589 2930;
  x(3)=-0.89260246649755573920606059 1127;
  x(4)=-0.80370495897252311568241745 5015;
  x(5)=-0.69168704306035320787489108 1289;
  x(6)=-0.55977083107394753460787154 8525;
  x(7)=-0.41175116146284264603593179 3833;
```

```
    x(8)=-0.25188622569150550958897285 4878;
    x(9)=-0.08477501304173530124226185 29358;
    x(10)= 0.08477501304173530124226185 29358;
    x(11)= 0.25188622569150550958897285 4878;
    x(12)= 0.41175116146284264603593179 3833;
    x(13)= 0.55977083107394753460787154 8525;
    x(14)= 0.69168704306035320787489108 1289;
    x(15)= 0.80370495897252311568241745 5015;
    x(16)= 0.89260246649755573920606059 1127;
    x(17)= 0.95582394957139775518119589 2930;
    x(18)= 0.99156516842093094673001600 4706;
  elseif (n == 19)
    x(1)=-0.99240684384358440318901767 0253;
    x(2)=-0.96020815213483003085277884 0688;
    x(3)=-0.90315590361481790164266092 8532;
    x(4)=-0.82271465653714282497892248 6713;
    x(5)=-0.72096617733522937861709586 0824;
    x(6)=-0.60054530466168102346963816 4946;
    x(7)=-0.46457074137596094571726714 8104;
    x(8)=-0.31656409996362983199011732 8850;
    x(9)=-0.16035864564022537586809611 5741;
    x(10)= 0;
    x(11)= 0.16035864564022537586809611 5741;
    x(12)= 0.31656409996362983199011732 8850;
    x(13)= 0.46457074137596094571726714 8104;
    x(14)= 0.60054530466168102346963816 4946;
    x(15)= 0.72096617733522937861709586 0824;
    x(16)= 0.82271465653714282497892248 6713;
    x(17)= 0.90315590361481790164266092 8532;
    x(18)= 0.96020815213483003085277884 0688;
    x(19)= 0.99240684384358440318901767 0253;
  elseif (n == 20)
    x(1)=-0.99312859918509492478612238 8471;
    x(2)=-0.96397192727791379126766613 1197;
    x(3)=-0.91223442825132590586775244 1203;
    x(4)=-0.83911697182221882339452906 1702;
    x(5)=-0.74633190646015079261430507 0356;
    x(6)=-0.63605368072651502545283669 6226;
    x(7)=-0.51086700195082709800436405 0955;
    x(8)=-0.37370608871541956067254817 7025;
    x(9)=-0.22778585114164507808049619 5369;
    x(10)=-0.07652652113349733375464040 93988;
    x(11)= 0.07652652113349733375464040 93988;
    x(12)= 0.22778585114164507808049619 5369;
    x(13)= 0.37370608871541956067254817 7025;
    x(14)= 0.51086700195082709800436405 0955;
    x(15)= 0.63605368072651502545283669 6226;
    x(16)= 0.74633190646015079261430507 0356;
    x(17)= 0.83911697182221882339452906 1702;
    x(18)= 0.91223442825132590586775244 1203;
    x(19)= 0.96397192727791379126766613 1197;
    x(20)= 0.99312859918509492478612238 8471;
  else
      error('GAUSS_WEIGHTS - Fatal error! Illegal value of n.')
  end
end

function w = gaussweights1d(n)
  w = ones(1,n);

  if ( n == 1 )
    w(1) = 2.0;
  elseif ( n == 2 )
    w(1) =  1.0;
    w(2) =  w(1);
  elseif ( n == 3 )
    w(1) =  0.55555555555555555555555555555565;
    w(2) =  0.88888888888888888888888888888889;
    w(3) =  0.55555555555555555555555555555565;
  elseif ( n == 4 )
```

```
  w(1) = 0.34785484513745385737306394922 2;
  w(2) = 0.65214515486254614262693605077 8;
  w(3) = 0.65214515486254614262693605077 8;
  w(4) = 0.34785484513745385737306394922 2;
elseif ( n == 5 )
  w(1) = 0.23692688505618908751426404072 0;
  w(2) = 0.47862867049936646804129151483 6;
  w(3) = 0.56888888888888888888888888888 9;
  w(4) = 0.47862867049936646804129151483 6;
  w(5) = 0.23692688505618908751426404072 0;
elseif ( n == 6 )
  w(1) = 0.17132449237917034504029614217 3;
  w(2) = 0.36076157304813860756983351383 8;
  w(3) = 0.46791393457269104738987034399 0;
  w(4) = 0.46791393457269104738987034399 0;
  w(5) = 0.36076157304813860756983351383 8;
  w(6) = 0.17132449237917034504029614217 3;
elseif ( n == 7 )
  w(1) = 0.12948496616886969327061143267 9;
  w(2) = 0.27970539148927666790146777142 4;
  w(3) = 0.38183005050511894495036977548 9;
  w(4) = 0.41795918367346938775510204081 6;
  w(5) = 0.38183005050511894495036977548 9;
  w(6) = 0.27970539148927666790146777142 4;
  w(7) = 0.12948496616886969327061143267 9;
elseif ( n == 8 )
  w(1) = 0.10122853629037625915253135431 0;
  w(2) = 0.22238103445337447054435599442 6;
  w(3) = 0.31370664587788728733796220198 7;
  w(4) = 0.36268378337836198296515044927 7;
  w(5) = 0.36268378337836198296515044927 7;
  w(6) = 0.31370664587788728733796220198 7;
  w(7) = 0.22238103445337447054435599442 6;
  w(8) = 0.10122853629037625915253135431 0;
elseif ( n == 9 )
  w(1) = 0.08127438836157441197189215811 05;
  w(2) = 0.18064816069485740405847203124 3;
  w(3) = 0.26061069640293546231874286941 9;
  w(4) = 0.31234707704000284006863040658 4;
  w(5) = 0.33023935500125976316452506928 7;
  w(6) = 0.31234707704000284006863040658 4;
  w(7) = 0.26061069640293546231874286941 9;
  w(8) = 0.18064816069485740405847203124 3;
  w(9) = 0.08127438836157441197189215811 05;
elseif ( n == 10 )
  w(1) =  0.06667134430868813759356880989 33;
  w(2) =  0.14945134915058059314577633965 8;
  w(3) =  0.21908636251598204399553493422 8;
  w(4) =  0.26926671930999635509122692156 9;
  w(5) =  0.29552422471475287017389299465 1;
  w(6) =  0.29552422471475287017389299465 1;
  w(7) =  0.26926671930999635509122692156 9;
  w(8) =  0.21908636251598204399553493422 8;
  w(9) =  0.14945134915058059314577633965 8;
  w(10) = 0.06667134430868813759356880989 33;
elseif ( n == 11 )
  w(1)= 0.05566856711617366648275372044 25;
  w(2)= 0.12558036946490462463469429922 4;
  w(3)= 0.18629021092773425142609764143 2;
  w(4)= 0.23319376459199047991852370484 3;
  w(5)= 0.26280454451024666218068886989 1;
  w(6)= 0.27292508677790063071448352833 6;
  w(7)= 0.26280454451024666218068886989 1;
  w(8)= 0.23319376459199047991852370484 3;
  w(9)= 0.18629021092773425142609764143 2;
  w(10)= 0.12558036946490462463469429922 4;
  w(11)= 0.05566856711617366648275372044 25;
elseif ( n == 12 )
  w(1)= 0.04717533638651182719461596148 50;
  w(2)= 0.10693932599531843096025471819 4;
```

```
  w(3)= 0.160078328543346226334652529543;
  w(4)= 0.203167426723065921749064455810;
  w(5)= 0.233492536538354808760849898925;
  w(6)= 0.249147045813402785000562436043;
  w(7)= 0.249147045813402785000562436043;
  w(8)= 0.233492536538354808760849898925;
  w(9)= 0.203167426723065921749064455810;
  w(10)= 0.160078328543346226334652529543;
  w(11)= 0.106939325995318430960254718194;
  w(12)= 0.047175336386511827194615961485O;
elseif ( n == 13 )
  w(1)= 0.040484004765315879520021592201O;
  w(2)= 0.092121499837728447914421775953S;
  w(3)= 0.138873510219787238463601776869;
  w(4)= 0.178145980761945738280046691996;
  w(5)= 0.207816047536888502312523219306;
  w(6)= 0.226283180262897238412090186040;
  w(7)= 0.232551553230873910194589515269;
  w(8)= 0.226283180262897238412090186040;
  w(9)= 0.207816047536888502312523219306;
  w(10)= 0.178145980761945738280046691996;
  w(11)= 0.138873510219787238463601776869;
  w(12)= 0.092121499837728447914421775953S;
  w(13)= 0.040484004765315879520021592201O;
elseif ( n == 14 )
  w(1)= 0.035119460331751863031832876138Z;
  w(2)= 0.080158087159760209805633277062S;
  w(3)= 0.121518570687903184689414809072;
  w(4)= 0.157203167158193534569601938624;
  w(5)= 0.185538397477937813741716590125;
  w(6)= 0.205198463721295603965924065661;
  w(7)= 0.215263853463157790195876443316;
  w(8)= 0.215263853463157790195876443316;
  w(9)= 0.205198463721295603965924065661;
  w(10)= 0.185538397477937813741716590125;
  w(11)= 0.157203167158193534569601938624;
  w(12)= 0.121518570687903184689414809072;
  w(13)= 0.080158087159760209805633277062S;
  w(14)= 0.035119460331751863031832876138Z;
elseif ( n == 15 )
  w(1)= 0.030753241996117268354628393577Z;
  w(2)= 0.070366047488108124709267416450S;
  w(3)= 0.107159220467171935011869546686;
  w(4)= 0.139570677926154314447804794511;
  w(5)= 0.166269205816993933553200860481;
  w(6)= 0.186161000015562211026800561866;
  w(7)= 0.198431485327111576456118326444;
  w(8)= 0.202578241925561272880620199968;
  w(9)= 0.198431485327111576456118326444;
  w(10)= 0.186161000015562211026800561866;
  w(11)= 0.166269205816993933553200860481;
  w(12)= 0.139570677926154314447804794511;
  w(13)= 0.107159220467171935011869546686;
  w(14)= 0.070366047488108124709267416450S;
  w(15)= 0.030753241996117268354628393577Z;
elseif ( n == 16 )
  w(1)= 0.027152459411754094851780572456O;
  w(2)= 0.062253523938647892862843836994O;
  w(3)= 0.095158511682492784809925107602Z;
  w(4)= 0.124628971255533872052476282192;
  w(5)= 0.149595988816576732081501730547;
  w(6)= 0.169156519395002538189312079030;
  w(7)= 0.182603415044923588866763667969;
  w(8)= 0.189450610455068496285396723208;
  w(9)= 0.189450610455068496285396723208;
  w(10)= 0.182603415044923588866763667969;
  w(11)= 0.169156519395002538189312079030;
  w(12)= 0.149595988816576732081501730547;
  w(13)= 0.124628971255533872052476282192;
  w(14)= 0.095158511682492784809925107602Z;
```

```
  w(15)= 0.062253523938647892862843836994 4;
  w(16)= 0.027152459411754094851780572456 0;
elseif ( n == 17 )
  w(1)= 0.024148302868547931960110026287 6;
  w(2)= 0.055459529373987201129440165358 2;
  w(3)= 0.085036148317179180883535370191 1;
  w(4)= 0.111883847193403971094788385626;
  w(5)= 0.135136368468525473286319981702;
  w(6)= 0.154045761076810288081431594802;
  w(7)= 0.168004102156450044509970663788;
  w(8)= 0.176562705366992646325270990113;
  w(9)= 0.179446470356206525458265644262;
  w(10)= 0.176562705366992646325270990113;
  w(11)= 0.168004102156450044509970663788;
  w(12)= 0.154045761076810288081431594802;
  w(13)= 0.135136368468525473286319981702;
  w(14)= 0.111883847193403971094788385626;
  w(15)= 0.085036148317179180883535370191 1;
  w(16)= 0.055459529373987201129440165358 2;
  w(17)= 0.024148302868547931960110026287 6;
elseif ( n == 18 )
  w(1)= 0.021616013526483310313342710266 5;
  w(2)= 0.049714548894969796453334946202 6;
  w(3)= 0.076425730254889056529129677616 6;
  w(4)= 0.100942044106287165562813984925;
  w(5)= 0.122555206711478460184519126800;
  w(6)= 0.140642914670650651204731303752;
  w(7)= 0.154684675126265244925418003836;
  w(8)= 0.164276483745832722986053776466;
  w(9)= 0.169142382963143591840656470135;
  w(10)= 0.169142382963143591840656470135;
  w(11)= 0.164276483745832722986053776466;
  w(12)= 0.154684675126265244925418003836;
  w(13)= 0.140642914670650651204731303752;
  w(14)= 0.122555206711478460184519126800;
  w(15)= 0.100942044106287165562813984925;
  w(16)= 0.076425730254889056529129677616 6;
  w(17)= 0.049714548894969796453334946202 6;
  w(18)= 0.021616013526483310313342710266 5;
elseif ( n == 19 )
  w(1)= 0.019461788229726477036312041464 4;
  w(2)= 0.044814226765699600332838157402 0;
  w(3)= 0.069044542737641226580708258006 0;
  w(4)= 0.091490021622449999464462094123 8;
  w(5)= 0.111566645547333994716023901682;
  w(6)= 0.128753962539336227675515784857;
  w(7)= 0.142606702173606611775746109442;
  w(8)= 0.152766042065859666778855400898;
  w(9)= 0.158968843393954347649956439465;
  w(10)= 0.161054449848783695979163625321;
  w(11)= 0.158968843393954347649956439465;
  w(12)= 0.152766042065859666778855400898;
  w(13)= 0.142606702173606611775746109442;
  w(14)= 0.128753962539336227675515784857;
  w(15)= 0.111566645547333994716023901682;
  w(16)= 0.091490021622449999464462094123 8;
  w(17)= 0.069044542737641226580708258006 0;
  w(18)= 0.044814226765699600332838157402 0;
  w(19)= 0.019461788229726477036312041464 4;
elseif ( n == 20 )
  w(1)= 0.017614007139152118311861962351 9;
  w(2)= 0.040601429800386941331039952274 9;
  w(3)= 0.062672048334109063569506535187 0;
  w(4)= 0.083276741576704748724758143222 0;
  w(5)= 0.101930119817240435036750135480;
  w(6)= 0.118194531961518417312377377711;
  w(7)= 0.131688638449176626898494499748;
  w(8)= 0.142096109318382051329298325067;
  w(9)= 0.149172986472603746787828737002;
  w(10)= 0.152753387130725850698084331955;
```

```
    w(11)= 0.152753387130725850698084331955;
    w(12)= 0.149172986472603746787828737002;
    w(13)= 0.142096109318382051329298325067;
    w(14)= 0.131688638449176626898494499748;
    w(15)= 0.118194531961518417312377377711;
    w(16)= 0.101930119817240435036750135480;
    w(17)= 0.083276741576704748724758143222 0;
    w(18)= 0.062672048334109063569506535187 0;
    w(19)= 0.040601429800386941331039952274 9;
    w(20)= 0.017614007139152118311861962351 9;
  else
    error('GAUSS_WEIGHTS - Fatal error! Illegal value of n.')
  end
end


end
```

```
clear all
clc;
pol=19;
insertU=50-1;
insertV=30-1;

filename=['C:\Users\user1\Desktop\Camila\Funciones\Datos\Data_',num2str(pol),'_',num2str(pol),'_'
,num2str(insertU),'x',num2str(insertV)];

% Data=load(filename); %clearvars pol
EXP_sign=[1 1 -1 1 1 1];
FEM_sign=[1 1 -1 1 1 -1 1];

save(filename,'EXP_sign','FEM_sign','-append');
```

```
function [ Mac ] = MAC( Exp, Num,lim )
% Compute Modal Assurance Criterion (MAC) matrix for experimental and
% numerical data.
%
%INPUT
% Exp   :Matrix containing experimental or FEM model data.
% Num   :Matrix containing numerical data.
%OUTPUT
% Mac   :Modal Assurance Criterion matrix.

Mac=zeros(lim);
Num=Mat2Vec(Num,lim);

for i=1:lim
    for j=1:lim
        Mac(i,j)=((Num(:,i)'*Exp(:,j))^2)/((Num(:,i)'*Num(:,i))*(Exp(:,i)'*Exp(:,i)));
    end
end

end

function [vector]=Mat2Vec(matriz,lim)
vector=zeros(size(matriz,1)*size(matriz,2),lim);
for k=1:lim
    for j=1:size(matriz,2)
        vector(9*(j-1)+1:9*(j-1)+9,k)=matriz(:,j,k);
    end
end

end
```

```
function [ xplot,yplot,Modes, Frec ] = Deflection( Data,index, knotU, knotV,L,xres,D,yres,w)
% Compute deflection for experimental measured points.
%
%INPUT
```

```
% Mod       :Matrix containing deflection for controlPoints.
% pU,V      :Polinomial degree for u and v directions.
% nU,V      :Number of elements for each dimention.
% knotU,V   :KnotVector for each direction.
% Tab       :Conectivity matrix.
%OUTPUT
% Modes     :Matrix containing deflection on desired points for desired
%           frecuencies.
xplot=linspace(0,L*(1-10e-5),xres);
yplot=linspace(0,D*(1-10e-5),yres);
Modes=zeros(length(yplot),length(xplot),size(index,2));

% Compute each deflection.
cont=1;
for num=index(1,:)
    for I=1:length(xplot)
        for J=1:length(yplot)
            %Find spans
            if nargin >8 && strcmp('c',w)
                i=FindSpan(Data.c_nU-1,Data.pU,xplot(I),knotU);
                j=FindSpan(Data.c_nV-1,Data.pV,yplot(J),knotV);
                mod=Data.Mod_c(FindIndex(i-Data.pU+1,j-Data.pV+1,Data.Tab),num);
            elseif nargin >8 && strcmp('w2',w)
                i=FindSpan(Data.w_nU-1,Data.pU,xplot(I),knotU);
                j=FindSpan(Data.w_nV-1,Data.pV,yplot(J),knotV);
                mod=Data.Mod_w2(FindIndex(i-Data.pU+1,j-Data.pV+1,Data.Tab),num);
            else
                i=FindSpan(Data.w_nU-1,Data.pU,xplot(I),knotU);
                j=FindSpan(Data.w_nV-1,Data.pV,yplot(J),knotV);
                mod=Data.Mod_w1(FindIndex(i-Data.pU+1,j-Data.pV+1,Data.Tab),num);
            end

            %Find controlPoints
            mod=reshape(mod,Data.pU+1,Data.pV+1);
            auxU=BasisFuns(xplot(I),Data.pU,knotU);
            auxV=BasisFuns(yplot(J),Data.pV,knotV);

            %Puntos a graficar
            Modes(J,I,cont)=auxU*mod*auxV';

        end
    end
    cont=cont+1;
end
Frec=Data.W(index(2,:));
end



function [ index ] = FindIndex( m,n,Table )
%Encuentra los indices asociados para una coordenada de elemento
%INPUT:
% m     :Element number, u direction.
% n     :Element number, v direction.
% Table :Conectivity indexes matrix.
%OUTPUT:
% index :Row vector containing the indexes.

for i=1:size(Table,1)
    if Table(i,1)==m && Table(i,2)==n
        index=Table(i,3:end);
        return
    end
end


end
```

```
function [ ModosEscalados ] = Escalar( Modos )
% Escala los modos.
```

```
if size(Modos,2)<=12,lim=size(Modos,2);
else lim=15;
end

maximos=max(Modos(:,1:lim));
minimos=min(Modos(:,1:lim));
p=maximos+abs(minimos);

ModosEscalados=zeros(size(Modos,1),lim);

for i=1:lim
    ModosEscalados(:,i)=(1/p(i))*Modos(:,i);
end

end
```

Anexo 2: Código: Pandeo de columna esbelta

2.1.-B-Splines

```
function [ Carga, Modos ] = Pand_Col_Esb_BSplines( L,h,t,un,E,p,insertionPoints,grafres )
% Calcula los modos de pandeo de para una columna esbelta, segun ecuación
% de Euler para condición de borde pasador-rodillo, utilizando método IGA
% con B-Splines. Grafica primeros 3 modos encontrados.
%
%INPUT:
% L                 :Largo de la columna, [m] o [inch]
% h                 :alto perfil (dirección de deformación), [m] o [inch]
% t                 :espesor perfil, [m] o [inch]
% un                :string con unidad de dimensiones, 'm' o 'inch'
% E                 :Modulo de Young del material, [Pa] o [psi]
% p                 :Grado polinomial de funciones de Base.
% insertionPoints   :# de puntos a insertar en vector de puntos.
% grafres           :# de puntos a considerar para graficar.
%OUTPUT:
% Carga             :Vector con cargas críticas de pandeo.
% Modos             :Matriz con modos calculados. Cada columna corresponde
%                   a las coordenadas de los puntos de control. Una columna
%                   por cada carga crítica calculada.

%=============Definir los parámetros de la viga============================
%Parámetros físicos de la columna:
I=(h^3*t)/12;

%Parámetros del modelo numérico: knotvector, grado polinomial, puntos de
%control.
knotVector=[zeros(1,p) linspace(0,L,insertionPoints+2) L*ones(1,p)]; %Se define con multiplicidad
p+1
n=length(knotVector)-p-1;
ControlPoints=linspace(0,L,n); %Se crean n puntos de control.

%==================Solución del Problema====================================

K_EI=zeros(n,n); %matrices para plantear el problema de valor propio
K_P=K_EI;
numel=insertionPoints+1;

for el=1:numel
   %Crea matrices elementales
   Ke_EI=zeros(p+1,p+1);
   Ke_P=Ke_EI;
   a=knotVector(el+p);
   b=knotVector(el+p+1);
   [GPoints,GWeigth]=GaussCuad( p,a,b );
   i=FindSpan(n-1,p,GPoints(1),knotVector);%Se encuentra el Span para ensamblar de manera más
sencilla.

   %Cálculo de la integral
   for sindex=1:length(GPoints);
       N=DerBasisFuns(GPoints(sindex),p,knotVector);
       J=LJacobian(i,N(2,:),p,ControlPoints);
       Ke_EI=Ke_EI+((b-a)/2)*E*I*(J\N(2,:)')*(J\N(2,:))*abs(J)*GWeigth(sindex);
       Ke_P=Ke_P+((b-a)/2)*(-1)*N(1,:)'*N(1,:)*abs(J)*GWeigth(sindex);
   end

   %Ensamble en la matriz global
   downlimit=i-p+1; uplimit=i+1;
   K_EI(downlimit:uplimit,downlimit:uplimit)=K_EI(downlimit:uplimit,downlimit:uplimit)+Ke_EI;
   K_P(downlimit:uplimit,downlimit:uplimit)=K_P(downlimit:uplimit,downlimit:uplimit)+Ke_P;
end

%Condiciones de borde: Doble empotrado
K_EI(1,1:n)=0;K_EI(1,1)=1;
K_P(1,1:n)=0;
```

```
K_EI(n,1:n)=0;K_EI(n,n)=1;
K_P(n,1:n)=0;

%Se resuelve la condición de los autovalores
[V,Y] = eig(K_EI,K_P);
Y=-1*Y;
[Carga,j]=sort(diag(Y));
Modos=V(:,j);

% ======================Graficar los modos================================
xplot=linspace(0,L*(1-10e-5),grafres);
yplot=zeros(size(xplot));

scrs=get(0,'screenSize');
figure('num','off','OuterPosition',[scrs(3)*0.20 scrs(4)*0.025 scrs(3)*0.6 scrs(4)*0.95])
set(gcf,'Color',[1 1 1])
num=1;
for index=1:n
    %Se buscan cargas no nulas
    if Carga(index)>0 && num<=3
        yplot=zeros(size(xplot));
        %Calcular para graficar
        for j=1:length(yplot)
            N=BasisFuns(xplot(j),p,knotVector);
            i=FindSpan(n-1,p,xplot(j),knotVector);
            downlimit=i-p+1;
            uplimit=i+1;
            yplot(j)=sum(N.*Modos(downlimit:uplimit,index)');
        end
        subplot(1,3,num)
        hold on
        plot([0 0],[0 L],'g:')
        plot(yplot,xplot)
        hold off
        title(['PCr_{',num2str(num),'}=',num2str(Carga(index),6)]);
        if num==1
            if min(yplot)<0, axis([min(yplot)*1.025 -min(yplot)*1.025 0 L]);
            else axis([-max(yplot)*1.025 max(yplot)*1.025 0 L]);
            end
        else axis([min(yplot)*1.025 max(yplot)*1.025 0 L]);
        end
        set(gca,'XTick',[]);
        ylabel(['x [',un,']']);
        xlabel('Deflexión');
        num=num+1;
    end
end

if p==1, mtit(['Pandeo de columna esbelta con B-Splines lineales y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
elseif p==2, mtit(['Pandeo de columna esbelta con B-Splines cuadráticas y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
elseif p==3, mtit(['Pandeo de columna esbelta con B-Splines cúbicas y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
else mtit(['Pandeo de columna esbelta con B-Splines de grado ',num2str(p),' y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
end

end
```

2.2.- NURBS

```
function [ Carga, Modos ] = Pand_Col_Esb_NURBS( L,h,t,un,E,p,insertionPoints,grafres )
% Calcula los modos de pandeo de para una columna esbelta, segun ecuación
% de Euler para condición de borde pasador-rodillo, utilizando método IGA
% con NURBS. Grafica primeros 3 modos encontrados.
%
%INPUT:
```

```
% L                 :Largo de la columna, [m] o [inch]
% h                 :alto perfil (dirección de deformación), [m] o [inch]
% t                 :espesor perfil, [m] o [inch]
% un                :string con unidad de dimensiones, 'm' o 'inch'
% E                 :Modulo de Young del material, [Pa] o [psi]
% p                 :Grado polinomial de funciones de Base.
% insertionPoints   :# de puntos a insertar en vector de puntos.
% grafres           :# de puntos a considerar para graficar.
%OUTPUT:
% Carga             :Vector con cargas críticas de pandeo.
% Modos             :Matriz con modos calculados. Cada columna corresponde
%                   a las coordenadas de los puntos de control. Una columna
%                   por cada carga crítica calculada.

%Parámetros físicos
I=(h^3*t)/12;

%Parámetros del modelo numérico: knotvector, grado polinomial, puntos de
%control.
knotVector=[zeros(1,p) linspace(0,L,insertionPoints+2) L*ones(1,p)]; %Se define con multiplicidad
p+1
n=length(knotVector)-p-1;
ControlPoints=linspace(0,L,n); %Se crean n puntos de control.
Weigths=ones(1,n); %En estas estructuras no se requiere pesos diferentes

%==================Solución del Problema=====================================

K_EI=zeros(n,n); %matrices para plantear el problema de valor propio
K_P=K_EI;
numel=insertionPoints+1;

for el=1:numel
   %Crea matrices elementales
   Ke_EI=zeros(p+1,p+1);
   Ke_P=Ke_EI;
   a=knotVector(el+p);
   b=knotVector(el+p+1);
   [GPoints,GWeigth]=GaussCuad( p,a,b );
   i=FindSpan(n-1,p,GPoints(1),knotVector);%Se encuentra el Span para ensamblar y ubicar peso
   downlimit=i-p+1; uplimit=i+1; %puntos de control asociados
   W=Weigths(downlimit:uplimit);

   %Cálculo de la integral
   for sindex=1:length(GPoints);
       aux=DerBasisFuns(GPoints(sindex),p,knotVector);
       N=aux(1,:).*W;
       DN=aux(2,:).*W;
       sN=sum(N);
       sDN=sum(DN);
       DN=(DN-sDN*sN*N)/sN;
       J=LJacobian(i,DN,p,ControlPoints);
       N=N/sN;
       Ke_EI=Ke_EI+((b-a)/2)*E*I*(J\DN')*(J\DN)*abs(J)*GWeigth(sindex);
       Ke_P=Ke_P+((b-a)/2)*(-1)*(N'*N)*abs(J)*GWeigth(sindex);
   end

   %Ensamble en la matriz global
   K_EI(downlimit:uplimit,downlimit:uplimit)=K_EI(downlimit:uplimit,downlimit:uplimit)+Ke_EI;
   K_P(downlimit:uplimit,downlimit:uplimit)=K_P(downlimit:uplimit,downlimit:uplimit)+Ke_P;
end

%Condiciones de borde: Doble empotrado
%Condiciones de borde: Doble empotrado
K_EI(1,1:n)=0;K_EI(1,1)=1;
K_P(1,1:n)=0;
K_EI(n,1:n)=0;K_EI(n,n)=1;
K_P(n,1:n)=0;

%Se resuelve la condición de los autovalores
[V,Y] = eig(K_EI,K_P);
```

```
Y=-1*Y;

[Carga,j]=sort(diag(Y));
Modos=V(:,j);

% ======================Graficar los modos=================================
xplot=linspace(0,L*(1-10e-5),grafres);
yplot=zeros(size(xplot));

scrs=get(0,'screenSize');
figure('num','off','OuterPosition',[scrs(3)*0.20 scrs(4)*0.025 scrs(3)*0.6 scrs(4)*0.95])
set(gcf,'Color',[1 1 1])
num=1;
for index=1:n
    %Se buscan cargas no nulas
    if Carga(index)>0 && num<=3
        yplot=zeros(size(xplot));
        %Calcular para graficar
        for j=1:length(yplot)
            N=BasisFuns(xplot(j),p,knotVector);
            N=N/sum(N);
            i=FindSpan(n-1,p,xplot(j),knotVector);
            downlimit=i-p+1;
            uplimit=i+1;
            yplot(j)=sum(N.*Modos(downlimit:uplimit,index)');
        end
        subplot(1,3,num)
        hold on
        plot([0 0],[0 L],'g:')
        plot(yplot,xplot)
        hold off
        title(['PCr_{',num2str(num),'}=',num2str(Carga(index),6)]);
        if num==1
            if min(yplot)<0, axis([min(yplot)*1.025 -min(yplot)*1.025 0 L]);
            else axis([-max(yplot)*1.025 max(yplot)*1.025 0 L]);
            end
        else axis([min(yplot)*1.025 max(yplot)*1.025 0 L]);
        end
        set(gca,'XTick',[]);
        ylabel(['x [',un,']']);
        xlabel('Deflexión');
        num=num+1;
    end
end


if p==1, mtit(['Pandeo de columna esbelta con NURBS lineales y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
elseif p==2, mtit(['Pandeo de columna esbelta con NURBS cuadráticas y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
elseif p==3, mtit(['Pandeo de columna esbelta con NURBS cúbicas y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
else mtit(['Pandeo de columna esbelta con NURBS de grado ',num2str(p),' y ',num2str(numel),'
elementos'],'fontsize',12,'color',[0 0 0],'xoff',0,'yoff',0.045);
end



end
```

Anexo 3: Código: modos de vibración de placa de Kirchhoff

3.1.- B-Splines

```
function [ K,M,nU,nV ] = Bending_plate_BSplines( L, pU, insertionPointsU,D, pV,
insertionPointsV,h,E,rho,Pois )
% Compute global stiffness and mass matrices for a plate of dimentions LxD,
% using isogeometric analysis with B-Splines basis functions.
%
%INPUT
% L,D               :Plate dimentions (width and high).
% pU,pV             :Polinomial degree of the B-Splines basis functions for
%                    each dimention.
% insertionPoints   :Points to insert on the knotVector for each dimention.
% h                 :Plate thickness.
% E                 :Young modulus of the plate material.
% rho               :Plate density.
% Pois              :Poisson ratio of the plate material.
%OUTPUT
% K                 :Stiffness matrix.
% M                 :Mass Matrix.
%
%NOTE: Use properties on SI  units (L, D and in [m], E in [Pa], rho in
%      [kg/m^3].

%=============Definir los parámetros de la placa============================
H=((E*h^3)/(12*(1-Pois^2)))*[1 Pois 0; Pois 1 0; 0 0 (1-Pois)/2];

%Parámetros del modelo numérico: U (dirección x), V (dirección y)
%en U
knotVectorU=[zeros(1,pU) linspace(0,L,insertionPointsU+2) L*ones(1,pU)]; %Se define con
multiplicidad p+1
nU=length(knotVectorU)-pU-1;
%En V
knotVectorV=[zeros(1,pV) linspace(0,D,insertionPointsV+2) D*ones(1,pV)]; %Se define con
multiplicidad p+1
nV=length(knotVectorV)-pV-1;

%Modelo
ControlPoints=zeros(nU*nV,2);
xbasis=linspace(0,L,nU)';
ybasis=linspace(0,D,nV);
for indexV=1:nV
    ControlPoints(nU*(indexV-1)+1:nU*(indexV-1)+nU,1)=xbasis;
    ControlPoints(nU*(indexV-1)+1:nU*(indexV-1)+nU,2)=ybasis(indexV)*ones(size(xbasis));
end
Tab=TableBending( insertionPointsU+1,insertionPointsV+1,pU,pV,nU);

%==================Solución del Problema====================================
K=zeros(nU*nV,nU*nV);
M=K;
knotU=unique(knotVectorU);
knotV=unique(knotVectorV);

for numelU=1:insertionPointsU+1
    for numelV=1:insertionPointsV+1
        aU=knotU(numelU);bU=knotU(numelU+1);%Intervalos de derivación
        aV=knotV(numelV);bV=knotV(numelV+1);
        [UGPoints,UGWeight]=GaussCuad(pU,aU,bU);
        [VGPoints,VGWeight]=GaussCuad(pV,aV,bV);
        Ke=zeros((pU+1)*(pV+1),(pU+1)*(pV+1));
        Me=Ke;
        i=FindSpan(nU,pU,UGPoints(1),knotVectorU);
        j=FindSpan(nV,pV,VGPoints(1),knotVectorV);
        %Se debe calcular la integral, por lo tanto, se recorren los puntos
        %de Gauss
        for ngpU=1:pU+1
            for ngpV=1:pV+1
```

```
                Be=zeros(3,(pU+1)*(pV+1));
                Ne1=zeros(1,(pU+1)*(pV+1));
                auxU=DerBasisFuns(UGPoints(ngpU),pU,knotVectorU,3);
                auxV=DerBasisFuns(VGPoints(ngpV),pV,knotVectorV,3);
                [ N,dNU,d2NU,dNV,d2NV,dNUdNV ]=derivadas(auxU,auxV);
                %Jacobiano primer orden
                J = BJacobian(i,j,nU,dNU,dNV,pU,pV,ControlPoints);
                %se reinica el contador
                cont=1;
                %Creación de vectores de funciones de forma
                for iV=1:pV+1
                    for iU=1:pU+1
                        Der2=J\[d2NU(iU,iV) dNUdNV(iU,iV); dNUdNV(iU,iV) d2NV(iU,iV)]/J;

                        Be(:,cont)=[ -Der2(1,1) ; -Der2(2,2) ; -2*Der2(2,1)];
                        Ne1(cont)= N(iU,iV);
                        cont=cont+1;
                    end
                end
                %Evaluar integración
                Ke=Ke+((bU-aU)/2)*((bV-
aV)/2)*(Be'*H*Be)*abs(det(J))*UGWeight(ngpU)*VGWeight(ngpV);
                Me=Me+((bU-aU)/2)*((bV-
aV)/2)*(Ne1'*rho*h*Ne1)*abs(det(J))*UGWeight(ngpU)*VGWeight(ngpV);
            end
        end
        %Ensamblar en la matriz global
        index=FindIndex(numelU,numelV,Tab);
        K(index,index)=K(index,index)+Ke;
        M(index,index)=M(index,index)+Me;

    end
end

end

function [ S,Su,Suu,Sv,Svv,Suv ]=derivadas(Nu,Nv)
% Compute NURBS surface derivatives for u,v point.

%Compute derivatives forms
S=Nu(1,:)'*Nv(1,:);
Su=Nu(2,:)'*Nv(1,:);
Sv=Nu(1,:)'*Nv(2,:);
Suu=Nu(3,:)'*Nv(1,:);
Svv=Nu(1,:)'*Nv(3,:);
Suv=Nu(2,:)'*Nv(2,:);

end

function [ Tab ] = TableBending( numelU,numelV,pU,pV,nU)
%Create a table to assemble.
Tab=zeros(numelU*numelV,(pU+1)*(pV+1)+2);
index=1;
for elU=1:numelU
    for elV=1:numelV
        Tab(index,1)=elU;
        Tab(index,2)=elV;
        for i=0:pU
            for j=0:pV
                Tab(index,(pU+1)*(j)+1+i+2)=nU*((elV+j)-1)+elU+i;
            end
        end
        index=index+1;
    end
end

end

function [ J ] = BJacobian(i,j,nU,DRU,DRV,pU,pV,P)
% Compute the Jacobian Matrix of a span for B-Spline functions.
```

```
%
% INPUT:
% i,i       :indexes of the Uspan and Vspan.
% nU        :number of Basis functions for u direction.
% DRU,DRV   :Derivatives on each direction.
% pU,pV     :Polinomial degrees.
% P         :Matrix of control points. It´s a (nU*nV)x2 matrix.
%OUTPUT:
% J         :Jacobian matrix.

J=zeros(2,2);
for indexU=1:pU+1
    for indexV=1:pV+1
        Upoint=i+indexU-pU;
        Vpoint=j+indexV-pV;
        Point=P(Upoint+nU*(Vpoint-1),:);
        J(:,1)=J(:,1)+DRU(indexU,indexV)*Point';
        J(:,2)=J(:,2)+DRV(indexU,indexV)*Point';
    end
end

end
```

3.2.- NURBS

Crear Matrices

```
function [ K,M,nU,nV,Tab ] = Bending_plate_NURBS( L, pU, insertionPointsU,D, pV,
insertionPointsV,h,E,rho,Pois )
% Compute global stiffness and mass matrices for a plate of dimentions LxD,
% using isogeometric analysis with NURBS basis functions.
%
%INPUT
% L,D               :Plate dimentions (width and high).
% pU,pV             :Polinomial degree of the B-Splines basis functions for
%                    each dimention.
% insertionPoints   :Points to insert on the knotVector for each dimention.
% h                 :Plate thickness.
% E                 :Young modulus of the plate material.
% rho               :Plate density.
% Pois              :Poisson ratio of the plate material.
%OUTPUT
% K                 :Stiffness matrix.
% M                 :Mass Matrix.
%
%NOTE: Use properties on SI  units (L, D and in [m], E in [Pa], rho in
%      [kg/m^3].

%=============Definir los parámetros de la placa============================
H=((E*h^3)/(12*(1-Pois^2)))*[1 Pois 0; Pois 1 0; 0 0 (1-Pois)/2];

%Parámetros del modelo numérico: U (dirección x), V (dirección y)
%en U
knotVectorU=[zeros(1,pU) linspace(0,L,insertionPointsU+2) L*ones(1,pU)]; %Se define con
multiplicidad p+1
nU=length(knotVectorU)-pU-1;
%En V
knotVectorV=[zeros(1,pV) linspace(0,D,insertionPointsV+2) D*ones(1,pV)]; %Se define con
multiplicidad p+1
nV=length(knotVectorV)-pV-1;

%Modelo
ControlPoints=zeros(nU*nV,2);
xbasis=linspace(0,L,nU)';
ybasis=linspace(0,D,nV);
for indexV=1:nV
    ControlPoints(nU*(indexV-1)+1:nU*(indexV-1)+nU,1)=xbasis;
```

```
    ControlPoints(nU*(indexV-1)+1:nU*(indexV-1)+nU,2)=ybasis(indexV)*ones(size(xbasis));
end
Tab=TableBending( insertionPointsU+1,insertionPointsV+1,pU,pV,nU);
weights=ones(1,nU*nV);

%==================Solución del Problema=====================================
K=zeros(nU*nV,nU*nV);
M=K;
knotU=unique(knotVectorU);
knotV=unique(knotVectorV);

for numelU=1:insertionPointsU+1
    for numelV=1:insertionPointsV+1
        %indices para encontrar pesos.
        index=FindIndex(numelU,numelV,Tab);
        Weight=reshape(weights(index),pU+1,pV+1);
        %Intervalos de derivación
        aU=knotU(numelU);bU=knotU(numelU+1);
        aV=knotV(numelV);bV=knotV(numelV+1);
        [UGPoints,UGWeight]=GaussCuad(pU,aU,bU);
        [VGPoints,VGWeight]=GaussCuad(pV,aV,bV);
        Ke=zeros((pU+1)*(pV+1),(pU+1)*(pV+1));
        Me=Ke;
        i=FindSpan(nU,pU,UGPoints(1),knotVectorU);
        j=FindSpan(nV,pV,VGPoints(1),knotVectorV);
        %Se debe calcular la integral, por lo tanto, se recorren los puntos
        %de Gauss
        for ngpU=1:pU+1
            for ngpV=1:pV+1
                Be=zeros(3,(pU+1)*(pV+1));
                Ne=zeros(1,(pU+1)*(pV+1));
                Nu=DerBasisFuns(UGPoints(ngpU),pU,knotVectorU,2);
                Nv=DerBasisFuns(VGPoints(ngpV),pV,knotVectorV,2);
                [ S,Su,Suu,Sv,Svv,Suv ]=derivadas(Nu,Nv,Weight);
                %Jacobiano primer orden
                J = BJacobian(i,j,nU,Su,Sv,pU,pV,ControlPoints);
                %se reinica el contador
                cont=1;
                %Creación de vectores de funciones de forma
                for iV=1:pV+1
                    for iU=1:pU+1
                        Der2=J\[Suu(iU,iV) Suv(iU,iV); Suv(iU,iV) Svv(iU,iV)]/J;
                        Be(:,cont)=[ -Der2(1,1) ; -Der2(2,2) ; -2*Der2(2,1)];
                        Ne(cont)= S(iU,iV);
                        cont=cont+1;
                    end
                end
                %Evaluar integración
                Ke=Ke+((bU-aU)/2)*((bV-
aV)/2)*(Be'*H*Be)*abs(det(J))*UGWeight(ngpU)*VGWeight(ngpV);
                Me=Me+((bU-aU)/2)*((bV-
aV)/2)*(Ne'*rho*h*Ne)*abs(det(J))*UGWeight(ngpU)*VGWeight(ngpV);
            end
        end
        %Ensamblar en la matriz global
        K(index,index)=K(index,index)+Ke;
        M(index,index)=M(index,index)+Me;

    end
end

end

function [ S,Su,Suu,Sv,Svv,Suv ]=derivadas(Nu,Nv,weights)
% Compute NURBS surface derivatives for u,v point.

%Compute all B-Splines derivatives, ponderate by weights and sums terms.
A=(Nu(1,:)'*Nv(1,:)).*weights; W=sum(sum(A));
Au=(Nu(2,:)'*Nv(1,:)).*weights; Wu=sum(sum(Au));
Auu=(Nu(3,:)'*Nv(1,:)).*weights; Wuu=sum(sum(Auu));
```

```
Av=(Nu(1,:)'*Nv(2,:)).*weights; Wv=sum(sum(Av));
Avv=(Nu(1,:)'*Nv(3,:)).*weights; Wvv=sum(sum(Avv));
Auv=(Nu(2,:)'*Nv(2,:)).*weights; Wuv=sum(sum(Auv));

%Compute derivatives forms
S=A/W;
Su=(Au-Wu*S)/W;
Suu=(Auu-2*Wu*Su-Wuu*S)/W;
Sv=(Av-Wv*S)/W;
Svv=(Avv-2*Wv*Sv-Wvv*S)/W;
Suv=(Auv-Wuv*S-Wu*Sv-Wv*Su)/W;

end

function [ Tab ] = TableBending( numelU,numelV,pU,pV,nU)
%Create a table to assemble.
Tab=zeros(numelU*numelV,(pU+1)*(pV+1)+2);
index=1;
for elU=1:numelU
    for elV=1:numelV
        Tab(index,1)=elU;
        Tab(index,2)=elV;
        for i=0:pU
            for j=0:pV
                Tab(index,(pU+1)*(j)+1+i+2)=nU*((elV+j)-1)+elU+i;
            end
        end
        index=index+1;
    end
end

end

function [ J ] = BJacobian(i,j,nU,DRU,DRV,pU,pV,P)
% Compute the Jacobian Matrix of a span for B-Spline functions.
%
% INPUT:
% i,i       :indexes of the Uspan and Vspan.
% nU        :number of Basis functions for u direction.
% DRU,DRV   :Derivatives on each direction.
% pU,pV     :Polinomial degrees.
% P         :Matrix of control points. It´s a (nU*nV)x2 matrix.
%OUTPUT:
% J         :Jacobian matrix.

J=zeros(2,2);
for indexU=1:pU+1
    for indexV=1:pV+1
        Upoint=i+indexU-pU;
        Vpoint=j+indexV-pV;
        Point=P(Upoint+nU*(Vpoint-1),:);
        J(:,1)=J(:,1)+DRU(indexU,indexV)*Point';
        J(:,2)=J(:,2)+DRV(indexU,indexV)*Point';
    end
end

end

function [ index ] = FindIndex( m,n,Table )
%Encuentra los indices asociados para una coordenada de elemento
%INPUT:
% m     :Element number, u direction.
% n     :Element number, v direction.
% Table :Conectiviti indexes matrix.
%OUTPUT:
% index :Row vector containing the indexes.

for i=1:size(Table,1)
    if Table(i,1)==m && Table(i,2)==n
        index=Table(i,3:end);
```

```
        return
    end
end


end
```

Resolver el problema

```
clear all
close all
clc;

%=======================Define material properties=========================

E=6.1372e10;
nu=0.33;
rho=2.5279e3;
h=8e-4;
L=0.35; %[m]
D=0.25; %[m]

%==================Create Stiffness and Mass matrices=====================
%Plate Layers
Data.pU=3;
insertionPointsU=34;
Data.pV=3;
insertionPointsV=24;
%===============Stiffness and mass matrices of the walls==================
[ K,M,Data.w_nU,Data.w_nV,Data.Tab ] = Bending_plate_NURBS( L, Data.pU, insertionPointsU,D,
Data.pV, insertionPointsV,h,E,rho,nu );

%================================Solve====================================

[Mod, Frec]=eig(K,M);
W_hz=sqrt(diag(Frec))/2/pi;
[W_hz,j]=sort(W_hz);
Data.W=abs(W_hz);
Data.Mod_w1=Mod(:,j); clearvars Mod W_hz Frec

%================================Graph====================================
Data.Mod_w1=Escalar(Data.Mod_w1);

knotU=[zeros(1,Data.pU) linspace(0,L,insertionPointsU+2) L*ones(1,Data.pU)]; %Se define con
multiplicidad p+1
knotV=[zeros(1,Data.pV) linspace(0,D,insertionPointsV+2) D*ones(1,Data.pV)]; %Se define con
multiplicidad q+1
xres=35;
yres=25;

%Compute deflection
[ xplot,yplot,Phi,Frec] = Deflection( Data,[4:15;4:15], knotU, knotV,L,xres,D,yres);

%Flip y axis:
auxN=zeros(size(Phi));
l=size(Phi(:,:,1),1);
for i=1:12
    for I=1:l
        auxN(I,:,i)=Phi(l+1-I,:,i);

    end
end
Phi=auxN;
clearvars auxN

%Graph
scrs=get(0,'ScreenSize');
s=[1 1 -1 -1 -1 -1 1 -1 -1 1 -1 1];
for n=1:12
```

```
    figure('Number','off','OuterPosition',[scrs(3)*0.25 scrs(4)*0.2 scrs(3)*0.5
scrs(4)*0.6],'Color',[1 1 1])
    mesh(xplot,yplot,s(n)*Phi(:,:,n))
    axis([0 0.35 0 0.25 -0.6 0.6])
    title(['\omega_{',num2str(n),'}=',num2str(Frec(n),6)],'fontsize',12)
    set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])
    set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])
    grid on
    campos([L -D/4 1])
end
```

Anexo 4: Código: Modos de vibración de panel compuesto tipo panal de abeja.

4.1.- Obtención de frecuencias naturales y modos normales.

```
%Solve the modal analysis of a honneycomb aluminum panel.
clear all
close all
clc

%=========================================================================
%===============Define material properties for all layers=================
%=========================================================================
%Walls
w_E=6.1372e10;
w_nu=0.33;
w_rho=2.5279e3;
w_h=8e-4;
%Core
c_E=2.4113e10;
c_nu=w_nu;
c_rho=31.1044;
c_h=10e-3;
%springs
k=8.2584e3; %[N/m]

%Define plate dimentions
L=0.35; %[m]
D=0.25; %[m]

%=========================================================================
%==================Create Stiffness and Mass matrices=====================
%=========================================================================

%Plate Layers
pU=19;
insertionPointsU=49;
pV=19;
insertionPointsV=29;
%===============Stiffness and mass matrices of the walls==================
[ w_K,w_M,w_nU,w_nV,Tab ] = Bending_plate_NURBS( L, pU, insertionPointsU,D, pV,
insertionPointsV,w_h,w_E,w_rho,w_nu );
%Core
[ c_K,c_M,c_nU,c_nV ] = Bending_plate_NURBS( L, pU, insertionPointsU,D, pV,
insertionPointsV,c_h,c_E,c_rho,c_nu );

%================Define global stiffnes and mass matrices=================
K=zeros(2*w_nU*w_nV+c_nU*c_nV);
M=K;

%============================Assemble layers==============================
%Indexes for each layer
w_1=(1:1:w_nU*w_nV);
c=(w_nU*w_nV+1:1:w_nU*w_nV+c_nU*c_nV);
w_2=(w_nU*w_nV+c_nU*c_nV+1:1:size(K,2));

%Wall_1
K(w_1,w_1)=w_K;
M(w_1,w_1)=w_M;
%Core
K(c,c)=c_K;
M(c,c)=c_M;
%Wall_2
K(w_2,w_2)=w_K;
M(w_2,w_2)=w_M;

%============================Add springs==================================
for i=1:length(w_1)
    %Wall 1-Core
```

```
    K([w_1(i) c(i)],[w_1(i) c(i)])= K([w_1(i) c(i)],[w_1(i) c(i)])+k*[1 -1;-1 1];
    %Core-Wall 2
    K([c(i) w_2(i)],[c(i) w_2(i)])=K([c(i) w_2(i)],[c(i) w_2(i)])+k*[1 -1;-1 1];
end

%=========================================================================
%================================Solve====================================
%=========================================================================

[Mod, Frec]=eig(K,M);
W_hz=sqrt(diag(Frec))/2/pi;
[W_hz,j]=sort(W_hz);
W_hz=abs(W_hz);
Mod=Mod(:,j);

% Result by layer
W=W_hz(4:23);
Mod_w1=Mod(w_1,:);
Mod_c=Mod(c,:);
Mod_w2=Mod(w_2,:);

%Save results
filename=['C:\Users\user1\Desktop\Camila\Funciones\Datos\Data_',num2str(pU),'_',num2str(pV),'_',n
um2str(insertionPointsU),'x',num2str(insertionPointsV),'.mat'];
save(filename,'W','Mod_w1','Mod_c','Mod_w2','pU','pV','insertionPointsU','insertionPointsV','w_nU
','w_nV','c_nU','c_nV','Tab');
```

4.2.- Correlación con modos experimentales.

```
function [ Wn, we, Mac ] = Comparar_modos_EXP( pol,numelU,numelV,graf )

insertU=numelU-1;
insertV=numelV-1;

Data=load(['C:\Users\user1\Desktop\Camila\Funciones\Datos\Data_',num2str(pol),'_',num2str(pol),'_
',num2str(insertU),'x',num2str(insertV)]); clearvars pol
load('C:\Users\user1\Desktop\Camila\Funciones\Datos\Phie');
load('C:\Users\user1\Desktop\Camila\Funciones\Datos\we');

%Define variables
L=0.35;
D=0.25;
knotU=[zeros(1,Data.pU) linspace(0,L,insertU+2) L*ones(1,Data.pU)]; %Se define con multiplicidad
p+1
knotV=[zeros(1,Data.pV) linspace(0,D,insertV+2) D*ones(1,Data.pV)]; %Se define con multiplicidad
q+1

%Escalar modos
Data.Mod_w1=Escalar(Data.Mod_w1);
Phie=Escalar(Phie);

%Calcular deflexión en puntos deseados
[ xplot,yplot,Phin, Wn ] = Deflection( Data,[[4:7,9:10];[1:4,6:7]], knotU, knotV,L,13,D,9);

%Voltear ejes:
auxN=zeros(size(Phin));
l=size(Phin(:,:,1),1);
for i=1:6
    for I=1:l
        auxN(I,:,i)=Phin(l+1-I,:,i);

    end
end
Phin=auxN;
clearvars auxN

%Correlación
```

```
Mac=diag(MAC(Phie,Phin,6));

%Graficar
if nargin>3 && strcmp('graficar',graf)
    scrs=get(0,'ScreenSize');
    for n=1:6
        Phin(:,:,n)=Data.EXP_sign(n)*Phin(:,:,n);

        figure('Number','off','Name',['Modo N°',num2str(n)],'OuterPosition',[scrs(3)*0.05
scrs(4)*0.2 scrs(3)*0.9 scrs(4)*0.6],'Color',[1 1 1])

        subplot(1,3,1)
        mesh(xplot,yplot,Phin(:,:,n))
        axis([0 0.35 0 0.25 -0.6 0.6])
        title(['\omega_{num}=',num2str(Wn(n),6)],'fontsize',12)
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])

        subplot(1,3,2)
        mesh(xplot,yplot,reshape(Phie(:,n),9,13))
        axis([0 0.35 0 0.25 -0.6 0.6])
        title(['\omega_{exp}=',num2str(we(n),6)],'fontsize',12)
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])

        subplot(1,3,3)
        hold on
        num=mesh(xplot,yplot,reshape(Phie(:,n),9,13));
        exp=mesh(xplot,yplot,Phin(:,:,n));
        hold off
        set(num, 'facecolor','none','edgecolor',[0 0 0]);
        set(exp, 'edgecolor',[1 0 0]);
        title(['MAC=',num2str(Mac(n))],'fontsize',12)
        axis([0 0.35 0 0.25 -0.6 0.6])
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])
        grid on
        legend('Experimental','Numerical','Location','SouthEast','Orientation','vertical')
        campos([0 -0 1])
    end

end

end
```

4.3.- Correlación con modos FEM

```
function [Mac] = Comparar_modos_FEM(pol,numelU,numelV,graf)

insertU=numelU-1;
insertV=numelV-1;

Data=load(['C:\Users\user1\Desktop\Camila\Funciones\Datos\Data_',num2str(pol),'_',num2str(pol),'_
',num2str(insertU),'x',num2str(insertV)]); %clearvars pol
load('C:\Users\user1\Desktop\Camila\Funciones\Datos\FEM_Data');

%Define variables
L=0.35;
D=0.25;
knotU=[zeros(1,Data.pU) linspace(0,L,insertU+2) L*ones(1,Data.pU)]; %Se define con multiplicidad
p+1
knotV=[zeros(1,Data.pV) linspace(0,D,insertV+2) D*ones(1,Data.pV)]; %Se define con multiplicidad
q+1

%Escalar modos
Data.Mod_w1=Escalar(Data.Mod_w1);
Phi=Escalar(Phi);

%Calcular deflexiones
[ xplot,yplot,Phin, Wn ] = Deflection( Data,[4:10;1:7], knotU, knotV,L,13,D,9);
```

```
%Voltear ejes:
auxN=zeros(size(Phin));
l=size(Phin(:,:,1),1);
for i=1:7
    for I=1:l
        auxN(I,:,i)=Phin(l+1-I,:,i);
    end
end
Phin=auxN;
clearvars auxN

%Correlación
Mac=diag(MAC(Phi,Phin,7));

%Graficar
if nargin>3 && strcmp('graficar',graf)
    scrs=get(0,'ScreenSize');
    for n=1:7
        Phin(:,:,n)=Data.FEM_sign(n)*Phin(:,:,n);

        figure('Number','off','Name',['Modo N°',num2str(n)],'OuterPosition',[scrs(3)*0.05
scrs(4)*0.15 scrs(3)*0.9 scrs(4)*0.7],'Color',[1 1 1])

        subplot(1,3,1)
        mesh(xplot,yplot,Phin(:,:,n))
        axis([0 0.35 0 0.25 -0.6 0.6])
        title(['\omega_{num}=',num2str(Wn(n),6)],'fontsize',12)
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])

        subplot(1,3,2)
        mesh(xplot,yplot,reshape(Phi(:,n),9,13))
        axis([0 0.35 0 0.25 -0.6 0.6])
        title(['\omega_{FEM}=',num2str(w(n),6)],'fontsize',12)
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])

        subplot(1,3,3)
        hold on
        num=mesh(xplot,yplot,reshape(Phi(:,n),9,13));
        exp=mesh(xplot,yplot,Phin(:,:,n));
        hold off
        set(num, 'facecolor','none','edgecolor',[0 0 0]);
        set(exp, 'edgecolor',[1 0 0]);
        title(['MAC=',num2str(Mac(n))],'fontsize',12)
        axis([0 0.35 0 0.25 -0.6 0.6])
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-0.6 0 0.6])
        grid on
        legend('FEM','IGA','Location','SouthEast','Orientation','vertical')
        campos([0 -0 1])

    end
end


end
```

4.4.- Gráfico de modos de panel compuesto.

```
function [] = GraphModes( pol, numelU, numelV ,resX,resY)
% GraphModes( Mod,W_hz, w_1,c,w_2,pUV,nUV,dim,knot,Tab,res)
insertU=numelU-1;
insertV=numelV-1;

Data=load(['C:\Users\user1\Desktop\Camila\Funciones\Datos\Data_',num2str(pol),'_',num2str(pol),'_
',num2str(insertU),'x',num2str(insertV)]); clearvars pol

%Define variables
L=0.35;
D=0.25;
```

```
knotU=[zeros(1,Data.pU) linspace(0,L,insertU+2) L*ones(1,Data.pU)]; %Se define con multiplicidad
p+1
knotV=[zeros(1,Data.pV) linspace(0,D,insertV+2) D*ones(1,Data.pV)]; %Se define con multiplicidad
q+1

%Calcular deflexión en puntos deseados
[ xplot,yplot,PhiW1 ] = Deflection( Data,[[4:7,9:10];[1:4,6:7]], knotU, knotV,L,resX,D,resY);
[ xplot,yplot,PhiC ] = Deflection( Data,[[4:7,9:10];[1:4,6:7]], knotU, knotV,L,resX,D,resY,'c');
[ xplot,yplot,PhiW2 ] = Deflection( Data,[[4:7,9:10];[1:4,6:7]], knotU,
knotV,L,resX,D,resY,'w2');


scrs=get(0,'ScreenSize');

for num=1:6
        figure('OuterPosition',[scrs(3)*0.05 scrs(4)*0.2 scrs(3)*0.9 scrs(4)*0.6],'Color',[1 1
1])
        %límites de gráficos
        max1=max(max(PhiW1(:,:,num)));min1=min(min(PhiW1(:,:,num))); l1=(max1+abs(min1))/2;
        max2=max(max(PhiC(:,:,num)));min2=min(min(PhiC(:,:,num))); %l2=(max(2)+abs(min2))/2;
        max3=max(max(PhiW2(:,:,num)));min3=min(min(PhiW2(:,:,num))); l3=(max3+abs(min3))/2;

        subplot(1,3,1)
        mesh(xplot,yplot,PhiW2(:,:,num)+(-5e-3-l3-abs(min2))*ones(size(PhiW2(:,:,num))))
        hold on
        mesh(xplot,yplot,PhiC(:,:,num))
        mesh(xplot,yplot,PhiW1(:,:,num)+(l1+5e-3+max2)*ones(size(PhiW1(:,:,num))))
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[0 3])
        grid on
        xlim([0 0.35])
        ylim([0 0.25])

        subplot(1,3,2)
        mesh(xplot,yplot,PhiW2(:,:,num)+(-5e-3-l3-abs(min2))*ones(size(PhiW2(:,:,num))))
        hold on
        mesh(xplot,yplot,PhiC(:,:,num))
        mesh(xplot,yplot,PhiW1(:,:,num)+(l1+5e-3+max2)*ones(size(PhiW1(:,:,num))))
        hold off
        campos([0.25/2 -20 0])
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-5e-3-l3-abs(min2) 0 l1+5e-3+max2])
        set(gca,'zticklabel',num2str(get(gca,'Ztick')',2))
        grid on
        xlim([0 0.35])
        ylim([0 0.25])

        subplot(1,3,3)
        mesh(xplot,yplot,PhiW2(:,:,num)+(-5e-3-l3-abs(min2))*ones(size(PhiW2(:,:,num))))
        hold on
        mesh(xplot,yplot,PhiC(:,:,num))
        mesh(xplot,yplot,PhiW1(:,:,num)+(l1+5e-3+max2)*ones(size(PhiW1(:,:,num))))
        hold off
        campos([20 0 0])
        set(gca,'Xtick',[0 0.35],'Ytick',[0 0.25],'Ztick',[-5e-3-l3-abs(min2) 0 l1+5e-3+max2] )
        set(gca,'zticklabel',num2str(get(gca,'Ztick')',2))
        grid on
        xlim([0 0.35])
        ylim([0 0.25])


end

end
```