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

OPTIMIZACIÓN TOPOLÓGICA EN PLACAS CUADRADAS Y COMPARACIÓN ENTRE ALGORITMOS DE OPTIMIZACIÓN MPGA Y MBPSO, APLICADOS A MÉTODOS IGA Y FEA

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

RAFAEL IGNACIO INOSTROZA AZÓCAR

PROFESORA GUÍA:
VIVIANA MERUANE NARANJO

MIEMBROS DE LA COMISIÓN:
RAFAEL RUIZ GARCÍA
ALEJANDRO ORTIZ BERNARDIN

Este trabajo ha sido parcialmente financiado por:
Núcleo Milenio en Metamateriales Mecánicos Suaves e Inteligentes

SANTIAGO DE CHILE
2022

RESUMEN DE: TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA Y MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: **RAFAEL IGNACIO INOSTROZA AZÓCAR**
FECHA: 2022
PROF. GUÍA: VIVIANA MERUANE NARANJO

OPTIMIZACIÓN TOPOLÓGICA EN PLACAS CUADRADAS Y COMPARACIÓN ENTRE ALGORITMOS DE OPTIMIZACIÓN MPGA Y MBPSO, APLICADOS A MÉTODOS IGA Y FEA

Es de vital importancia el estudio de tópicos relacionados a la aislación de vibraciones y cómo estas juegan un rol en confiabilidad y mantenimiento de equipos mecánicos o estructurales que estén sometidos a excitaciones periódicas. En este contexto, es importante el estudio de placas que cumplan estos roles, las cuales son modeladas con los métodos FEA e IGA, donde este último busca disminuir la brecha entre los métodos CAD y FEA, reemplazando las funciones de forma utilizadas en FEA por las utilizadas en software CAD, conocidas como NURBS.

Los algoritmos de optimización MpGA (Multipopulation Genetic Algorithm) y MBPSO (Modified Binary Particle Swarm Optimization) pertenecen a la familia de algoritmos evolutivos, los cuales consisten en heurísticas que usan una población, la cual evoluciona o se desplaza por el espacio solución según reglas predefinidas y en base a una función de objetivo la cual asigna un valor numérico al desempeño de cada individuo según el problema a resolver, cantidad conocida como fitness.

El objetivo de este trabajo consiste en determinar la aplicabilidad del algoritmo de optimización MBPSO para resolver problemas de optimización topológica en placas cuadradas modeladas usando los métodos IGA y FEA, y su comparación al método MPGA.

La metodología de este trabajo consiste optimizar placas cuadradas para todas las combinaciones entre modelos y algoritmos de optimización, seguido de la optimización de placas cuadradas con orificios circulares para la mejor combinación resultante del caso anterior. En ambos casos se reportan y estudian las topologías resultantes, los diagramas de bandas, tamaño del bandgap y curvas de convergencia.

Como principales conclusiones se tiene que la combinación de IGA y MBPSO es la que arroja mejores resultados, ya que esta combinación asegura la convergencia mas rápida y los mejores resultados promedio. Al momento de optimizar placas con orificios se obtiene que agregar restricciones a la geometría ayuda a la convergencia del algoritmo, además se obtienen bandgaps mayores a medida que se aumenta el radio del orificio. Para ambos estudios se obtiene que los mejores resultados corresponden a los entregados por placas con 32 elementos por arista.

Agradecimientos

Este trabajo de tesis ha requerido mucho esfuerzo y dedicación por mi parte. Ha sido un largo camino, que, desde mis inicios como estudiante, siempre he sentido el apoyo incondicional de mi familia, quienes estuvieron siempre allí para ayudarme en todo lo que necesitaba. Gracias por hacer grandes esfuerzos para que viniese a estudiar y desarrollar mi vida de la mejor forma posible en Santiago. Gracias por enseñarme a trabajar duro y a dar lo mejor de mi cada día. Por estar siempre ahí, por apoyarme en mis decisiones y por celebrar mis alegrías como si fuesen suyas.

De forma especial, quisiera agradecer a mi madre por fundar en mí grandes valores, como la responsabilidad y perseverancia, los que han sido fundamentales para lograr este trabajo y estar a puertas de mi titulación. A pesar de la distancia, siempre sentí su apoyo y amor, y lo valoro profundamente.

A mi hermano Carlos, por ser un modelo a seguir, por acompañarme y decirme siempre que puedo, por darme los consejos precisos en los momentos claves, por mostrarme que no existen imposibles y por alentarme a ser cada día mejor, a aspirar a mayores y mejores cosas.

En esta oportunidad también quisiera honrar a mi abuela Edith, que siempre alimentó con su experiencia mis pasos, fomentó mis hábitos de estudio y siempre me dio amor y sabios consejos. Gracias infinitas para esta gran mujer, que siempre, desde que fui un infante, me cuidó y amó, hasta los días de hoy.

A mis amigos del colegio y universidad, quieres me acompañaron en esta travesía de diferentes maneras, a pesar de desaparecerme por un tiempo y no ser tan activo socialmente.

Por último, pero no menos importante, agradezco a mi profesora de tesis Viviana Meruane por orientarme en este difícil camino, sin sus directrices, no hubiese podido concluir con este excelente trabajo.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

La optimización de estructuras ha sido un tópico relevante para la ingeniería a lo largo del tiempo, primero se desarrollaron técnicas de optimización de tamaño y de forma, para luego dar lugar a la optimización topológica, la cual ha sido desarrollada enormemente desde su primera aparición, propuesta por Bendsøe y Kikuchi en 1988 [1]. La optimización topológica consiste en encontrar la mejor forma de disponer un material en una región del espacio con el fin de obtener el mejor desempeño posible (bajo algún criterio preestablecido), cumpliendo además con ciertas condiciones de diseño. Inicialmente, este método fue concebido para optimizar diseño de estructuras, pero fue rápidamente usado y adaptado a otras áreas, tales como fluidos, óptica, acústica, entre otras.

Unas de las aplicaciones de la optimización topológica es la maximización de *bandgaps* en estructuras periódicas, es decir, la maximización del rango de frecuencias por el cual una onda no se puede propagar por este tipo de estructuras. Se espera que en el futuro este tipo de materiales tengan aplicaciones en procesamiento de señales acústicas, filtros acústicos, resonadores, entre otras.

Para realizar la optimización se debe usar algún tipo de algoritmo que maximice (o minimice) una función objetivo, cambiando una serie de variables que definan el problema y el sistema físico en el cual se basa, cumpliendo además una serie de restricciones previamente definidas. Los algoritmos usados para optimización topológica se pueden dividir en dos grupos, primero se encuentran los algoritmos basados en gradiente, que reciben su nombre gracias a que se deben calcular derivadas o gradientes, de la función objetivo en cada iteración, con el fin de determinar la dirección de cambio de las variables de optimización en el paso siguiente del algoritmo. El segundo grupo los constituyen los algoritmos evolutivos, que se caracterizan por tener una población que evoluciona o cambia de acuerdo a ciertas reglas de selección, reproducción o evolución. En este trabajo de tesis se utilizan algoritmos evolutivos, principalmente porque se trabaja con variables en un espacio binario y se desconoce un punto en específico donde se partirá la optimización del algoritmo.

Particularmente, se trabaja con dos tipos de algoritmos evolutivos, estos son: Algoritmos genéticos multipoblacional (MpGA) y Optimización Binaria Modificada por enjambre de partículas (MBPSO), los cuales se diferencian principalmente por el método usado para generar la población en el siguiente paso del algoritmo. El primero cuenta con varias poblaciones que se desarrollan de forma aislada y en paralelo, donde cada una tiene padres que se re-

producen con cierta regla de selección para generar la siguiente población. Se cuenta además con métodos para generar una mayor variabilidad en la población resultante, estos son: la mutación y el entrecruzamiento. Por su parte, el segundo algoritmo cuenta con un método donde los individuos se desplazan por el espacio solución en una dirección promedio entre el mejor individuo de la iteración anterior y el mejor individuo encontrado hasta el momento.

Para generar una geometría se pueden usar diferentes métodos. El primero es el de elementos finitos (FEM), el cual destaca por ser el método más usado en diferentes áreas de la ingeniería y en una serie de programas comerciales, tales como: Ansys, Adina, Comsol, entre otros. Pese a lo anterior, su mayor problema radica en su dificultad para trabajar con geometrías complejas. Los softwares CAD (Computer Aided Design), tales como AutoCad, Inventor, entre otros, utilizan funciones NURBS (Non Uniform Rational B-splines) para generar cualquier tipo de geometrías o curvas. FEM recibe un modelo CAD y crea una malla, la cual es una aproximación de la geometría real creada por las funciones NURBS. El mallado tiene problemas de precisión para geometrías complejas, lo que genera a su vez errores en los cálculos realizados, especialmente en zonas de concentración de esfuerzos.

Para subsanar el problema antes mencionado, Hughes et al. [2] propone el método IGA (Isogeometric Analysis), ya que utiliza funciones NURBS tanto para la creación de la geometría como para el cálculo del campo escalar que se desea determinar. Con el paso del tiempo IGA sea ha ido convirtiendo en una herramienta esencial para el análisis numérico, encontrando además aplicaciones en otras áreas de la ingeniería y las ciencias, tales como óptica, fluidos, electromagnetismo, entre otras.

En este trabajo, se evalúa la implementación del algoritmo de optimización MBPSO para la resolución de problemas de optimización topológica con el fin de optimizar el bandgap resultante del diagrama de bandas en placas cuadradas, se comparan los resultados referentes a tiempos de ejecución, número de iteraciones hasta la convergencia, topologías resultantes en cada caso y los valores de bandgaps para cada topología con los entregados por el algoritmo MpGA. Las placas son modeladas con los métodos FEA e IGA.

1.1. Motivación

En ingeniería mecánica, la optimización topológica es un método de optimización de estructuras donde se busca mejorar el diseño de un componente mecánico o estructural, contando con parámetros constantes, tales como: tipo de material, cargas aplicadas, condiciones de borde, entre otras. Estos parámetros son dependientes del problema a resolver y varían de acuerdo a cada caso. En optimización topológica se optimiza la disposición del material en una geometría predefinida, donde cada sección de la misma puede ser retirado o conservado, de tal forma que se maximice el desempeño en una función en particular, a diferencia de otros métodos de optimización de estructuras como optimización de tamaño o de forma, donde el diseño de la geometría se mantiene intacto.

A lo largo del tiempo, la optimización topológica ha sido una herramienta usada para discriminar y determinar el mejor concepto en etapas tempranas de diseño. También se han usado para aligerar estructuras, mientras cumplan con las condiciones de diseño impuestas para industrias como la automotriz, aeroespacial, construcción, entre otras. Lo anterior re-

percute en que el diseño final sea más liviano y necesite una menor cantidad de material para su fabricación, lo que se traduce en una importante reducción de costos de manufactura y transporte, principalmente. Un ejemplo de lo anterior es la utilización de la optimización para la reducción del peso en un 84 % de soportes de motores, lo que llevó a un ahorro de 31 millones de dolares para la empresa General Electric en el año 2016 [3].

Se espera que este trabajo sea un aporte en el marco del diseño de estructuras, contribuyendo con una metodología de estudio y comparación de los resultados. También se extiende la información referente al uso de los algoritmos de optimización usados, como estos se comportan y como solucionan un mismo problema bajo las mismas condiciones y usando los mismos parámetros, para que de esta forma se pueda discriminar entre estos en futuras optimizaciones.

En particular, en este trabajo, se demuestra la viabilidad del uso del algoritmo MBPSO y sus ventajas por sobre el algormitmo MpGA, donde se demuestra que en todos los casos de estudio presenta menores tiempos de optimización y mayores resultados promedios.

1.2. Objetivos

A continuación se presentan los objetivos generales y específicos del proyecto de tesis.

1.2.1. Objetivo General

- El objetivo general del proyecto de tesis consiste en determinar la aplicabilidad del algoritmo de optimización MBPSO para resolver problemas de optimización topológica en placas cuadradas modeladas usando los métodos IGA y FEA, y su comparación al método MPGA.

1.2.2. Objetivos Específicos

- Validar los modelos de placas cuadradas utilizando Comsol Multiphysics.
- Realizar optimización topológica utilizando los algoritmos MBPSO y MpGA aplicados a placas modeladas con FEA e IGA.
- Comparación de resultados de optimización topológica entre algoritmos de optimización aplciados a los diferentes modelos para generar las placas.
- Optimización topológica de placa con orificio circular, utilizando algoritmo de optimización con mejores resutlados.

Capítulo 2

Marco teórico

El objetivo de este capítulo es mostrar una revisión de la literatura y fuente consultadas, haciendo énfasis en el marco teórico y antecedentes que sustenta este trabajo.

2.1. Antecedentes

2.1.1. Antecedentes de estudio

Optimización de estructuras

La optimización de estructuras ha sido un tópico relevante para la ingeniería a lo largo del tiempo, la que consiste principalmente en optimizar una serie de parámetros o características que definen a una estructura, de tal forma que cumpla ciertas condiciones de diseño. Se usan diferentes métodos para modelar las estructuras y diferentes algoritmos de optimización. Existen 3 tipos de optimización en estructuras mecánicas, estas son:

- Optimización de forma: Este método consiste en optimizar la forma que adopta el perímetro la estructura o sub-estructura de trabajo [4]. Se debe definir de antemano la geometría a optimizar, por ejemplo, si se desea optimizar agujeros, estos deben ser definidos a priori, no creados desde cero durante la optimización. La optimización de forma ha sido usada ampliamente a lo largo del tiempo en trabajos tales como la optimización de la forma de alas de aviones [5].

- Optimización topológica: Este método trata de determinar carácteristicas de una geometría tales como el número, ubicación y forma de agujeros. En estos procedimientos se cuenta con un dominio constante en todo momento, permitiendo generar agujeros a partir de una geometrìa continua. Este tipo de optimización ha sido usada en casos tales como la optimización de la topología de materiales compuestos [6].

- Optimización de tamaño: Este método de optimización trata de ajustar parámetros tales como espesor, largo, ancho o profundidad del material, manteniendo la forma y la topología constantes [4]. La optimización de tamaño ha sido vital en el diseño de estructuras complejas que distribuyan sus esfuerzos internos de manera eficiente [7].

FEA

FEA o *Finite element analysis* es un método de simulaciones numérica basado en FEM o *Finite element method*. En el marco de la ingeniería mecánica se ha ido desarrollando a partir desde principios del siglo XX [8], con trabajos pioneros en el área, tales como el de W. Ritz [9], donde se trabaja por primera vez con el principio variacional aplicado a placas estáticas con condiciones de borde de empotramiento. Galerkin, en 1915 [10] introduce por primera vez el cálculo de coeficiente indeterminados de funciones de prueba mediante relaciones de ortogonalidad entre ellas.

En las siguiente décadas se consolida el método de elementos finitos, hasta tal punto que se desarrollan programas que automatizan el análisis de problemas tanto estáticos como dinámicos, creando programas basados en FORTRAN y en ALGOL, como lo señala Buck et al. [11]. Actualmente, existen una gran variedad de programas comerciales que trabajan con elementos finitos, tales como ANSYS o COMSOL y librerias para diferentes lenguajes de programación, tales como MFEM, destinada a C++, o SDTools, usada en Matlab, la cual ha sido utilizada en investigaciones relacionadas a temas tales como vibración de materiales viscoelásticos [12, 13], chirridos y vibración no lineal [14, 15] o maquinaria rotativa [16, 17].

IGA

IGA o *Isogeometric analysis* es un método de análisis numérico introducido por Hughes et al. [2] el cual consiste en la creación de una base de funciones denominadas NURBS (Non-Uniform Rational B-Splines), las cuales son también utilizadas como base en los estudios CAD, siendo su principal objetivo el reducir la brecha existente entre éste tipo de estudios con los propios de ingeniería.

IGA surge como una alternativa al uso de FEM, dado que, a diferencia de este último, IGA usa la misma base para representar tanto la geometría como la solución al problema en cuestión, tal y como lo señala Hughes et al. [2].

A lo largo del tiempo se han propuesto e implementado una serie de extenciones a IGA, para de esta forma poder resolver una mayor cantidad de problemas, entre los que se encuentran el método XIGA, que modela la fractura en materiales [18, 19]. Además existen métodos de mallado adaptativo en función de diferentes medidas para el error [20, 21, 22].

GA

GA o *Genetic Algorithm* es un método de optimización heurístico basado en la mecánica de la selección natural en seres vivos. En algoritmos genéticos se crean individuos (que componen una población) cuyo material genéticos es representación de los parámetros que definen al modelo a optimizar. Una población evoluciona en el tiempo, donde los individuos pueden mutar o entrecruzarse para generar descendencia.

Los algoritmos genéticos han sido usados ampliamente en el área de la ingeniería mecánica, encontrando soluciones a problemas tales como la optimización de fibras para la creación de materiales compuestos [23], optimización de intercambiadores de calor de placas con orificios [24] o la optimización de la planificación de la ruta de robots automatizados [25].

MBPSO

MBPSO o *Modified Binary Particle Swarm Optimization*, propuesta por Lee et al. [26], es una modificación al método BPSO o *Binary Particle Swarm Optimization*, propuesta por Kennedy et al. [27] y corresponde a la versión binaria de PSO o *Particle Swarm Optimization*, propuesta también por el autor anterior [28]. Ésta última técnica es una heurística basada o motivada en el comportamiento y movimiento de organismos vivos. PSO y sus derivados, a diferencia de GA, cuentan con dos funciones que se actualizan en cada iteración del algoritmo, éstas son la velocidad y la posición de cada individuo.

La familia de algoritmos de PSO han sido utilizados en el área de la ingeniería mecánica, encontrando soluciones a problemas tales como el diseño de rodamientos híbridos [29], en el diseño y optimización de tolerancias de mecanizado [30] y en la optimización de los parámetros de operación de una cortadora láser para obtener la rugosidad óptima en el corte de una hoja acrílica [31].

2.1.2. Forma débil de la ecuación diferencial

Sea una placa plana delgada modelada según la teoría de Kirchhoff-Love [32], se consideran coordenadas cartesianas, donde la normal a la superficie apunta en la dirección z. Se considera también la ecuación diferencial que gobierna el problema de las vibraciones de una placa en ausencia de fuerzas externas.

$$D\nabla^4\mathbf{u} + \rho\frac{\partial^2\mathbf{u}}{\partial t^2} = 0 \tag{2.1}$$

donde ∇^4 es el operador diferencial biarmónico, o sea $\nabla^4 = \nabla^2\nabla^2$ y ∇^2 es el operador laplaciano, ρ es la densidad del material en cuestión y D es la rigidez a la flexión, definida como sigue:

$$D = \frac{Eh^3}{12(1-\nu^2)} \tag{2.2}$$

De la ecuación anterior se tiene que E es el módulo de elasticidad, h es el espesor y ν es el coeficiente de Poisson de la placa.

Dado que sólo la deflexión en el eje z es una variable independiente, se establece la siguiente relación para la deflexión $\mathbf{u}$:

$$\mathbf{u} = \begin{bmatrix} u & v & w \end{bmatrix}^T = \begin{bmatrix} -z\frac{\partial}{\partial x} & -z\frac{\partial}{\partial y} & 1 \end{bmatrix}^T = \mathbf{T}w \tag{2.3}$$

Se define también la ecuación constitutiva para placas delgadas, donde se considera el caso de esfuerzo plano.

$$\boldsymbol{\sigma}_p = \mathbf{D}\boldsymbol{\epsilon}_p \tag{2.4}$$

donde $\mathbf{D}$ es una matriz que depende de las propiedades de la placa y se define de la siguiente manera para placas homogéneas:

$$\mathbf{D} = D \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{2.5}$$

El pseudo-esfuerzo y la pseudo-deformación $\boldsymbol{\sigma}_p$ y $\boldsymbol{\epsilon}_p$ respectivamente, son definidas como sigue:

$$\boldsymbol{\epsilon}_p = \begin{bmatrix} -\frac{\partial^2}{\partial x^2} & -\frac{\partial^2}{\partial y^2} & -2\frac{\partial^2}{\partial x \partial y} \end{bmatrix}^T w = \mathbf{L}w \tag{2.6}$$

$$\boldsymbol{\sigma}_p = \begin{bmatrix} M_x & M_y & M_{xy} \end{bmatrix}^T \tag{2.7}$$

M_x, M_y y M_{xy} son componentes del momento de flexión y torsión respectivamente, definidas por la siguiente ecuación:

$$\begin{aligned} M_x &= -D(\frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2}) \\ M_y &= -D(\frac{\nu \partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}) \\ M_{xy} &= -D(1-\nu)(\frac{\nu \partial^2 w}{\partial x \partial y}) \end{aligned} \tag{2.8}$$

Para una placa libre, la forma débil de Galerkin de la ecuación diferencial de la ecuación 2.1 es la que se muestra a continuación, lo anterior es equivalente a aplicar el principio variacional o el de trabajos virtuales.

$$\int_\Omega \delta \boldsymbol{\epsilon}_p^T \boldsymbol{\sigma}_p d\Omega + \int_\Omega \delta \mathbf{u}^T \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} d\Omega = 0 \tag{2.9}$$

Si se reemplazan las ecuaciones 2.3, 2.4 y 2.6 en la ecuación 2.9, se obtiene la ecuación 2.10, cuya expresión es la ecuación final a resolver, tanto para IGA como para FEA.

$$\int_\Omega \delta(\mathbf{L}w)^T \mathbf{D}(\mathbf{L}w) d\Omega + \int_\Omega \delta(\mathbf{T}w)^T \rho(\mathbf{T}\frac{\partial^2 w}{\partial t^2}) d\Omega = 0 \tag{2.10}$$

2.1.3. FEA

En esta sección se presentan los conceptos necesarios para entender cómo el método de elementos finitos discretiza una solución a una ecuación.

2.1.3.1. Discretización de la solución

La siguiente ecuación describe como se discretiza el desplazamiento [33]:

$$\mathbf{u^h}(\mathbf{x}, \mathbf{y}) = \sum_i^{n_d} N_i \mathbf{u}(\mathbf{x_i}, \mathbf{y_i}) \tag{2.11}$$

donde $\mathbf{u^h}(\mathbf{x}, \mathbf{y})$ es la aproximación del campo de desplazamientos según elementos finitos, $\mathbf{N_i}$ es una matriz con los valores de las funciones de forma para el nodo i, n_d es el número de

nodos del sistema, n_f es el número de grados de libertad por nodo, dado lo anterior, $\mathbf{N_i}$ tiene la siguiente forma matricial:

$$N_i = \begin{bmatrix} N_{i,1} & 0 & 0 & 0 \\ 0 & N_{i,2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & N_{i,n_f} \end{bmatrix} \tag{2.12}$$

Se sustituye la ecuación 2.11 en la ecuación 2.10 y se obtiene la ecuación dinámica final en su forma discretizada para el caso de elementos finitos.

$$\mathbf{M\ddot{w}} + \mathbf{Kw} = 0 \tag{2.13}$$

Con $\mathbf{M}$ y $\mathbf{K}$ las matrices de masa y rigidez respectivamente, definidas como sigue:

$$\mathbf{M}_{I,J} = \int_\Omega (\rho \phi_I \phi_J h \phi + \phi_{I,x}\phi_{J,x}\frac{h^3}{12} + \phi_{I,y}\phi_{J,y}\frac{h^3}{12}) d\Omega \tag{2.14}$$

$$\mathbf{K}_{I,J} = \int_\Omega (\mathbf{B}_I^T \mathbf{D} \mathbf{B}_J) d\Omega \tag{2.15}$$

$$\mathbf{B}_I = \begin{bmatrix} -\phi_{I,xx} & -\phi_{I,yy} & -2\phi_{I,xy} \end{bmatrix}^T \tag{2.16}$$

donde $\phi_I = N_i$, el segundo índice de esta nueva notación corresponde a la variable con respecto a la que se está derivando.

2.1.4. IGA

En esta sección se presentan los conceptos necesarios para entender como el método IGA discretiza una solución a una ecuación.

2.1.4.1. Discretización de la solución

En este trabajo la base de funciones NURBS es empleada tanto en la parametrización de la geometría, como en la discretización de la solución. El dominio físico se denota como Ω y el dominio en el espacio paramétrico se denomina por $\hat{\Omega}$, donde este último es el dominio que contiene al vector *knot*, para hacer un mapeo entre el espacio paramétrico y el espacio físico, se sigue la siguiente transformación [34]:

$$\mathbf{X} = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q}(\boldsymbol{\xi})\mathbf{T_{i,j}} \tag{2.17}$$

donde $\boldsymbol{\xi} = (\xi, \eta)$ son las coordenadas paramétricas del sistema de coordenadas definido por las NURBS, $\mathbf{X} = (x, y)$ son las coordenadas en el sistema físico, $R_{i,j}^{p,q}$ es la base de funciones NURBS y $\mathbf{T_{i,j}}$ son los puntos de control.

Para determinar los desplazamientos se tiene la siguiente ecuación:

$$\mathbf{u} = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q}(\boldsymbol{\xi})\mathbf{u_{i,j}} \tag{2.18}$$

donde $\mathbf{u_{i,j}}$ son las variables de control y $\mathbf{u}$ es el desplazamiento real.

Se sustituye la ecuación 2.18 en la ecuación 2.10 y se obtiene la ecuación dinámica final en su forma discretizada para el caso de IGA.

$$\mathbf{M}\ddot{\mathbf{w}} + \mathbf{K}\mathbf{w} = 0 \tag{2.19}$$

Con $\mathbf{M}$ y $\mathbf{K}$ las matrices de masa y rigidez respectivamente, definidas de la siguiente manera:

$$\mathbf{M}_{I,J} = \int_\Omega (\rho\phi_I\phi_J h\phi + \phi_{I,x}\phi_{J,x}\frac{h^3}{12} + \phi_{I,y}\phi_{J,y}\frac{h^3}{12})d\Omega \tag{2.20}$$

$$\mathbf{K}_{I,J} = \int_\Omega (\mathbf{B}_I^T\mathbf{D}\mathbf{B}_J)d\Omega \tag{2.21}$$

$$\mathbf{B}_I = \begin{bmatrix} -\phi_{I,xx} & -\phi_{I,yy} & -2\phi_{I,xy} \end{bmatrix}^T \tag{2.22}$$

Siguiendo la misma notación mostrada en la sección 2.1.3.1, $\phi_I = R_{i,j}^{p,q}$ y $I = 1, 2, \ldots, n\times m$, el segundo índice de esta nueva notación corresponde a la variable con respecto a la que se está derivando.

2.1.5. Condiciones de borde periódicas

Las condiciones de borde se definen luego de haber definido las ecuaciones de movimiento que gobiernan una celda unitaria libre de fuerzas externas, se procede a la aplicación de las condiciones de borde periódicas. Se usan soluciones de onda plana según la ecuación A.17 y se reemplaza en la ecuación 2.13 o a la ecuación 2.19, obteniéndose la ecuación 2.23.

$$(\mathbf{K} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{D}\mathbf{q} = 0, \quad \mathbf{D} = \mathbf{K} - \omega^2\mathbf{M} \tag{2.23}$$

donde $\mathbf{D}$ se conoce como la rigidez dinámica y se reduce a la rigidez estática para frecuencias cero. Se cumple que $\mathbf{D}$ es una matriz real y semi-definida positiva.

Luego, usando el teorema de Bloch, se puede establecer una relación entre los desplazamientos en los bordes de la celda primitiva, para esto se hace la distinción entre FEA e IGA.

Los procedimientos siguientes están basado en lo mostrado mostrado en [35], pero adaptado a las características de las placas usadas para los métodos FEA e IGA, respectivamente.

FEA

Primero recordar que, dada la formulación del problema en elementos finitos, un desplazamiento cualquiera de la celda $\mathbf{q_j}$ tiene 3 componentes, una de desplazamiento en el eje z y dos componentes relacionadas a rotaciones, asociadas a los ejes x e y, luego $\mathbf{q_j} = \begin{bmatrix} z_j & \theta_{x,j} & \theta_{y,j} \end{bmatrix}^T$.

Teniendo en consideración lo anterior se procede a usar el teorema de Bloch en los bordes de la celda, resultando las ecuaciones mostradas a continuación.

$$\begin{aligned} \mathbf{q}_{tl} = e^{k_2}\mathbf{q}_{bl}, \quad \mathbf{q}_{tr} = e^{k_1+k_2}\mathbf{q}_{bl}, \quad \mathbf{q}_{br} = e^{k_1}\mathbf{q}_{bl} \\ \mathbf{q}_r = e^{k_1}\mathbf{q}_l, \quad \mathbf{q}_t = e^{k_2}\mathbf{q}_b \end{aligned} \tag{2.24}$$

donde los sub índices t, b, l, r, i denotan los desplazamientos asociados a los puntos superiores (top), inferiores (bottom), izquierdos (left), derechos (right) e internos (internal), respectivamente.

Para simplificar la notación se decide dotar al vector $\mathbf{q_j}$ de dos índices inferiores y un índice superior, donde los dos inferiores hacen referencia al número del nodo según las coordenadas x e y de la celda, donde se cuenta desde abajo hacia arriba y desde izquierda a derecha, mientras que el incide superior hace referencia al número de celda, donde se decide que la celda unitaria de referencia tiene asignado el número 1, por ejemplo $\mathbf{q^1_{1,1}}$ y $\mathbf{q^1_{n,n}}$ hacen referencia al nodo de la esquina inferior izquierda y al nodo de la esquina superior derecha de la celda unitaria de referencia respectivamente.

Las relaciones de la ecuación 2.24 se pueden resumir en una matriz de transformación $\mathbf{T}$, cumpliendo la siguiente relación:

$$\mathbf{q}^i = \mathbf{T}\tilde{\mathbf{q}}^\mathbf{i} \tag{2.25}$$

donde $\tilde{\mathbf{q}}^\mathbf{i}$ son los desplazamientos en el espacio reducido de Bloch, el espacio al cual se le aplican las condiciones de borde periódicas del problema. $\mathbf{T}$ se define como sigue:

$$\mathbf{T_1} = \begin{bmatrix} I_{3\times3} & 0 & \cdots & 0 \\ 0 & I_{3\times3} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_{3\times3} \\ e^{k_2}I_{3\times3} & 0 & \cdots & 0 \end{bmatrix} \tag{2.26}$$

$$\mathbf{T_2} = \begin{bmatrix} e^{k_1}I_{3\times3} & 0 & \cdots & 0 \\ 0 & e^{k_1}I_{3\times3} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{k_1}I_{3\times3} \\ e^{k_1+k_2}I_{3\times3} & 0 & \cdots & 0 \end{bmatrix} = e^{k_1}\mathbf{T_1} \tag{2.27}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}\otimes\mathbf{T_1} \\ \begin{bmatrix}\mathbf{T_2} & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix} \end{bmatrix} \tag{2.28}$$

donde $\otimes$ corresponde al producto de kronecker entre matrices. $\mathbf{T_1}$ y contribución a la matriz de transformación $\mathbf{T}$ muestran la aplicación de las condiciones de borde en toda la celda, excepto las del borde derecho de la misma, donde $\mathbf{T_2}$ cumple esta función. Dada la forma en que se define $\mathbf{T}$ y el ordenamiento de sus componentes, $\tilde{\mathbf{q}}^\mathbf{i}$ queda definida según la siguiente ecuación:

$$\tilde{\mathbf{q}}^\mathbf{i} = \begin{bmatrix}\widetilde{\mathbf{q_1}}^\mathbf{i} & \widetilde{\mathbf{q_2}}^\mathbf{i} & \widetilde{\mathbf{q_3}}^\mathbf{i} & \cdots & \widetilde{\mathbf{q_{n-1}}}^\mathbf{i}\end{bmatrix}^T \tag{2.29}$$

donde $\tilde{\mathbf{q}}_\mathbf{j}$ es un vector de los desplazamientos comprendidos en una columna de la celda, donde se cuenta desde abajo hacia arriba para cada $\tilde{\mathbf{q}}_\mathbf{j}$ y desde izquierda a derecha para cada valor de j.

$$\tilde{\mathbf{q}}_\mathbf{j}^\mathbf{i} = \begin{bmatrix} \mathbf{q}_{\mathbf{j},\mathbf{1}}^\mathbf{i} & \mathbf{q}_{\mathbf{j},\mathbf{2}}^\mathbf{i} & \cdots & \mathbf{q}_{\mathbf{j},\mathbf{n-1}}^\mathbf{i} \end{bmatrix}^T \tag{2.30}$$

IGA

Se realizan definiciones análogas de las matrices y relaciones de la sección anterior, pero adaptadas a IGA, donde se decide mantener la misma notación para $\mathbf{q}$. Se tiene que, dadas las definiciones hechas en la sección 2.1.4, $\mathbf{q}$ solo representa los desplazamientos en el eje z, pero dado que los bordes está representado por dos filas o columnas de puntos, se debe establecer también una relación entre la primera derivada de los desplazamientos en el borde de la celda. Para finalizar, se aplica el teorema de Bloch a la relación encontrada.

Se hace la derivación para un par nodos de un borde arbitrario, dado que el procedimiento es general y puede ser aplicado de forma indistinta en cualquier par de nodos del borde, el procedimiento es el siguiente:

Se igualan las diferencias entre los desplazamientos en un borde, lo cual es equivalente a igualar las derivadas, dado que la distancia entre cada punto es la misma.

$$\mathbf{q}_{\mathbf{1,n-1}}^\mathbf{1} - \mathbf{q}_{\mathbf{1,n}}^\mathbf{1} = \mathbf{q}_{\mathbf{1,1}}^\mathbf{2} - \mathbf{q}_{\mathbf{1,2}}^\mathbf{2} \tag{2.31}$$

Se considera que $\mathbf{q}_{\mathbf{1,n}}^\mathbf{1} = \mathbf{q}_{\mathbf{1,1}}^\mathbf{2}$ y que $\mathbf{q}_{\mathbf{1,2}}^\mathbf{1} = \mathbf{q}_{\mathbf{1,2}}^\mathbf{2}$, entonces:

$$\mathbf{q}_{\mathbf{1,n-1}}^\mathbf{1} - \mathbf{q}_{\mathbf{1,n}}^\mathbf{1} = \mathbf{q}_{\mathbf{1,n}}^\mathbf{1} - \mathbf{q}_{\mathbf{1,2}}^\mathbf{1} \tag{2.32}$$

Ahora se puede encontrar una ecuación para el valor de $\mathbf{q}_{\mathbf{1,n-1}}^\mathbf{1}$.

$$\mathbf{q}_{\mathbf{1,n-1}}^\mathbf{1} = 2\mathbf{q}_{\mathbf{1,n}}^\mathbf{1} - \mathbf{q}_{\mathbf{1,2}}^\mathbf{1} = 2\mathbf{q}_{\mathbf{1,1}}^\mathbf{1} - \mathbf{q}_{\mathbf{1,2}}^\mathbf{1} \tag{2.33}$$

Con esto se pueden escribir la versiones para IGA de la matriz de transformación:

$$\mathbf{T_1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 2e^{k_2} & -e^{k_2} & 0 & \cdots & 0 \\ e^{k_2} & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.34}$$

$$\mathbf{T_2} = e^{k_1}\mathbf{T_1} \tag{2.35}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} \otimes \mathbf{T_1} \\ \begin{bmatrix} 2 & -1 & 0 \cdots 0 \\ 1 & 0 & 0 \cdots 0 \end{bmatrix} \otimes \mathbf{T_2} \end{bmatrix} \tag{2.36}$$

De esta forma, la matriz de transformación $\mathbf{T}$ queda completamente definida, luego $\tilde{\mathbf{q}}^{\mathbf{i}}$ para el caso IGA se define de la misma manera y con el mismo ordenamiento que para FEA, cumpliéndose nuevamente la ecuación 2.25.

2.1.6. Formulación

Teniendo en consideración todo lo definido y mostrado en la sección anterior se procede a resolver el sistema de la ecuación 2.23.

$$\mathbf{Dq} = 0 \tag{2.37}$$

Se considera el cambio de variables definido por la matriz de transformación $\mathbf{T}$ (el cual es independiente de si se trabaja con IGA o FEA), reemplazando la ecuación 2.25 en la ecuación anterior, resultando la siguiente expresión:

$$\mathbf{DT}\tilde{\mathbf{q}} = 0 \tag{2.38}$$

Si se pre multiplica por la conjugada transpuesta de $\mathbf{T}$, o sea $\mathbf{T}^{\dagger}$, y si se hace la definición $\widetilde{\mathbf{D}} \equiv \mathbf{T}^{\dagger}\mathbf{DT}$ se obtiene la ecuación de valores y vectores propios a resolver.

$$\widetilde{\mathbf{D}}\tilde{\mathbf{q}} = 0 \tag{2.39}$$

Recordando que se debe resolver el sistema para diferentes valores de $\mathbf{k}$, los cuales varían según el contorno de la primera zona irreductible de Brillouin definida en la sección A.4, se tiene que la obtención de los autovalores y autovectores son dependientes de $\mathbf{k}$, lo que es equivalente a definirlos en función de k_1 y de k_2. Luego, la ecuación de valores y vectores propios se escribe como sigue:

$$\widetilde{\mathbf{D}}(k_1, k_2, \omega)\tilde{\mathbf{q}} = 0 \tag{2.40}$$

Dado que la matriz $\widetilde{\mathbf{D}}$ es semi-definida positiva se cumple que sus autovalores (ω^2) son reales y no nulos y por ende los valores de las frecuencias naturales también lo son.

2.1.7. MpGA

Los algoritmos genéticos multipoblacionales o MpGA, buscan mejorar el desempeño de los algoritmos genéticos tradicionales, ya que se cuenta con poblaciones que se desarrollan por separado y en paralelo, las cuales sólo comparten individuos cuando todas ellas detienen su evolución. En particular, cada una comparte con el resto el individuo con el mejor desempeño, reemplazando así a los peores individuos de cada una de la otras poblaciones.

Luego de compartir individuos entre poblaciones, éstas se aíslan nuevamente y comienza otro proceso de optimización. Los ciclos de optimización y de compartir información terminan cuando se cumple un siguiente criterio, dónde ϵ_2 es el parámetro de término de la optimización e i, j corresponden a las poblaciones del algoritmo.

$$Si \quad \sum_{j,j\neq i} \|max(Fitness_i) - max(Fitness_j)\| < \epsilon_2, \quad \forall i \Rightarrow Parar \tag{2.41}$$

A continuación, se presentan diagramas de como operan los algoritmos genéticos, donde el primero muestra el funcionamiento de una población en particular y el segundo del algoritmo en su totalidad considerando solo dos poblaciones en paralelo. La extensión a más poblaciones es trivial.

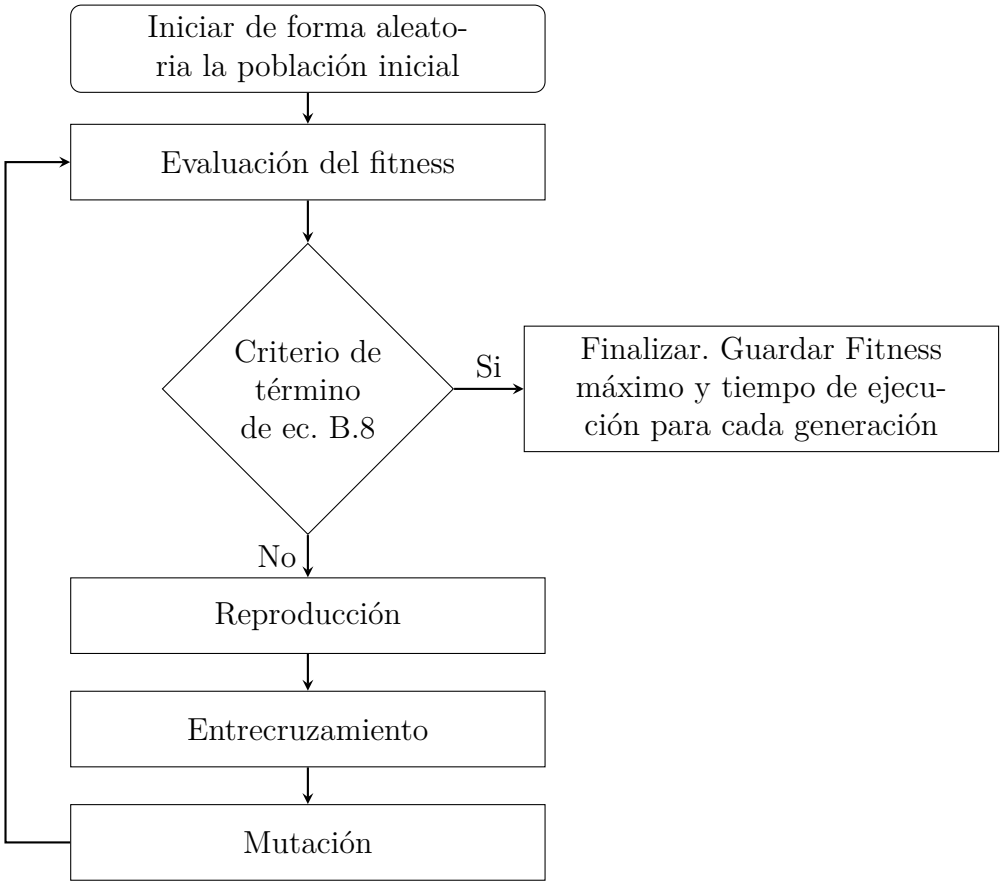


Figura 2.1: Diagrama una población en algoritmos genéticos.

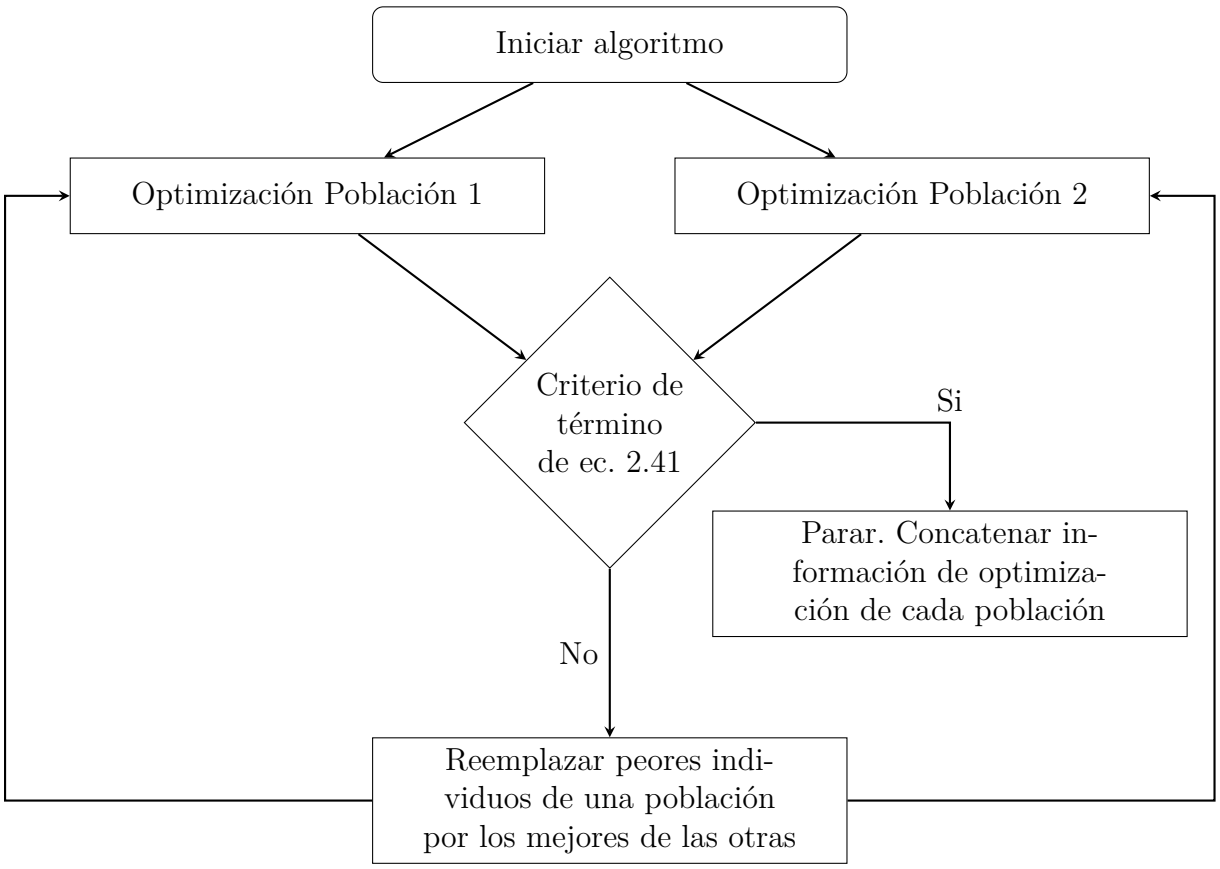


Figura 2.2: Diagrama de un algoritmo genético MpGA con dos poblaciones.

2.1.8. MBPSO

MBPSO surge como la modificación que busca resolver el problema de BPSO, es decir, busca que el algoritmo no se atrape o estanque en los mínimos locales del problema [26]. Para lograr este objetivo, se propone el siguiente set de ecuaciones:

$$v_i^k(t+1) = wv_i^k(t) + c_1u_1(p_{best,i} - x_i^k(t)) + c_2u_1(g_{best} - x_i^k(t)) \tag{2.42}$$

$$x_i^k(t+1) = x_i^k(t) + v_i^k(t+1) \tag{2.43}$$

$$T(x_i^k(t)) = \frac{1}{1+e^{-x_i^k(t)}} \tag{2.44}$$

$$x_i^k(t+1) = \begin{cases} 0 & si \quad u < T(x_i^k(t+1)) \\ 1 & si \quad u \geq T(x_i^k(t+1)) \end{cases} \tag{2.45}$$

$$Fitness_i(t) = F(x_i^1(t), x_i^2(t)), \ldots, x_i^n(t)) \tag{2.46}$$

De lo anterior se tiene que la velocidad se usa para calcular la posición en una etapa intermedia del algoritmo. Luego esta posición ser usada como variable en la función de transferencia para el cálculo del umbral y su posterior uso en el cálculo de la posición final Cabe destacar que los primeros dos pasos del algoritmo son los usados en PSO y luego se usan

los pasos del método 2 de BPSO con la salvedad de usar la posición $x_i^k(t+1)$ en vez de la velocidad en la función de transferencia. En este caso, el fitness $Fitness_i(t)$ se calcula usando todos los valores de cada componente de la posición de cada partícula, donde n es el número de componentes, o sea es la dimensión del espacio.

La modificación antes mencionada solo resuelve en parte el problema de BPSO, por lo que se hace una última modificación a las ecuaciones 2.44 y 2.45, la cual consiste en cambiar la función usada para definir $T(x)$. Para ello, se definen funciones divididas en dos familias diferentes, la familia de funciones con forma de S y la familia de funciones con forma de V [36]. Esta última modificación consiste en cambiar el cómo se calcula el valor final de la posición, que depende del tipo de familia usado. A continuación se detallan diferentes cambios.

$$T(x) = \frac{1}{1+e^{-ax}}, \quad a \geq 0 \tag{2.47}$$

$$\begin{aligned} T(x) = |\operatorname{erf}(x\frac{\sqrt{\pi}}{2})| = |\frac{\sqrt{\pi}}{2}\int_0^{\frac{\sqrt{\pi}}{2}x} e^{-t^2}dt| \\ T(x) = |\tanh(x)| \\ T(x) = \frac{x}{\sqrt{x^2+1}} \\ T(x) = |\frac{2}{\pi}\arctan\Big(\frac{2}{\pi}\Big)x| \end{aligned} \tag{2.48}$$

$$x_i^k(t+1) = \begin{cases} 1 - x_i^k(t) & si \quad u < T(v_i^k(t+1)) \\ x_i^k(t) & si \quad u \geq T(v_i^k(t+1)) \end{cases} \tag{2.49}$$

La ecuación 2.47 corresponde a la familia S de funciones de transferencia, mientras que la ecuación 2.48 corresponde a la familia V de funciones de transferencia. La diferencia entre las funciones dentro de una misma familia radica principalmente en como éstas se comportan según los cambios en la variable x, es decir, qué tan sensibles son a estos cambios..

La ecuación 2.49 corresponde a la modificación del cálculo de $x_i^k(t+1)$ para el caso de funciones con forma de V. Para el caso de la familia de funciones con forma de S, la función no sufre modificaciones, y por ende se usa la ecuación 2.45.

A continuación se muestran ejemplos de funciones de transferencia tipo S y V, donde se aprecia diferencias notables en cómo responden a cambios en el valor de x.

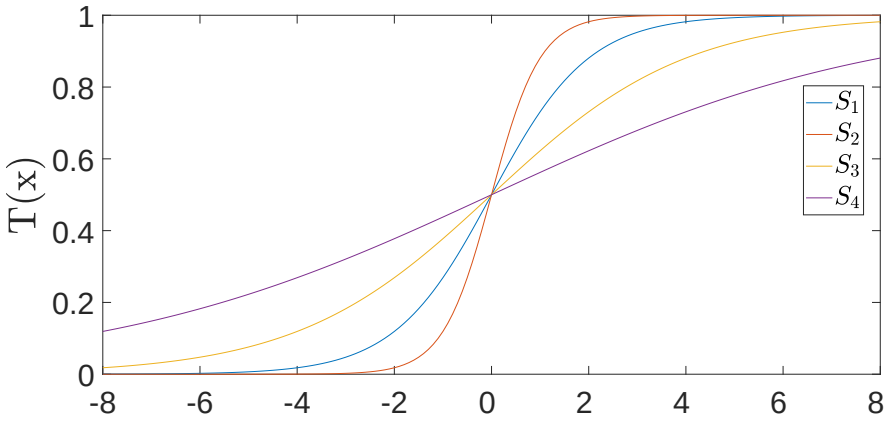


Figura 2.3: Familia de funciones de transferencia tipo S, donde S_i utiliza los valores de a $a = 1, 2, \frac{1}{2}, \frac{1}{4}$, respectivamente.

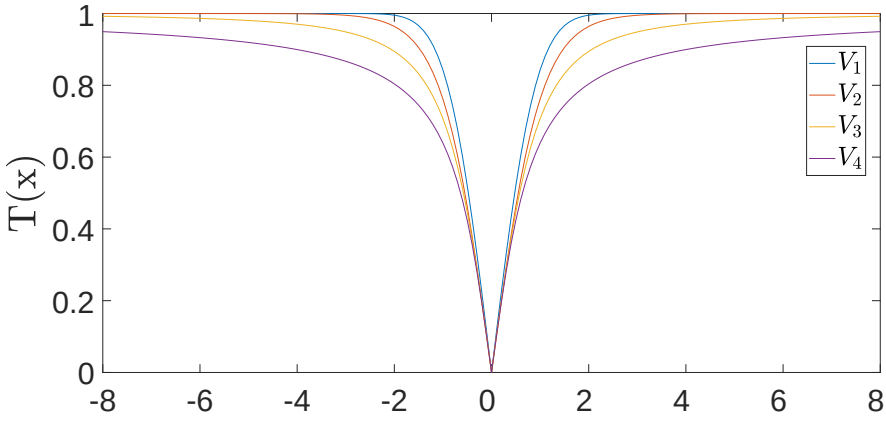


Figura 2.4: Familia de funciones de transferencia tipo V, donde V_i son las funciones de la ecuación 2.48.

A continuación se presenta una diagrama de como opera éste algoritmo durante la optimización.

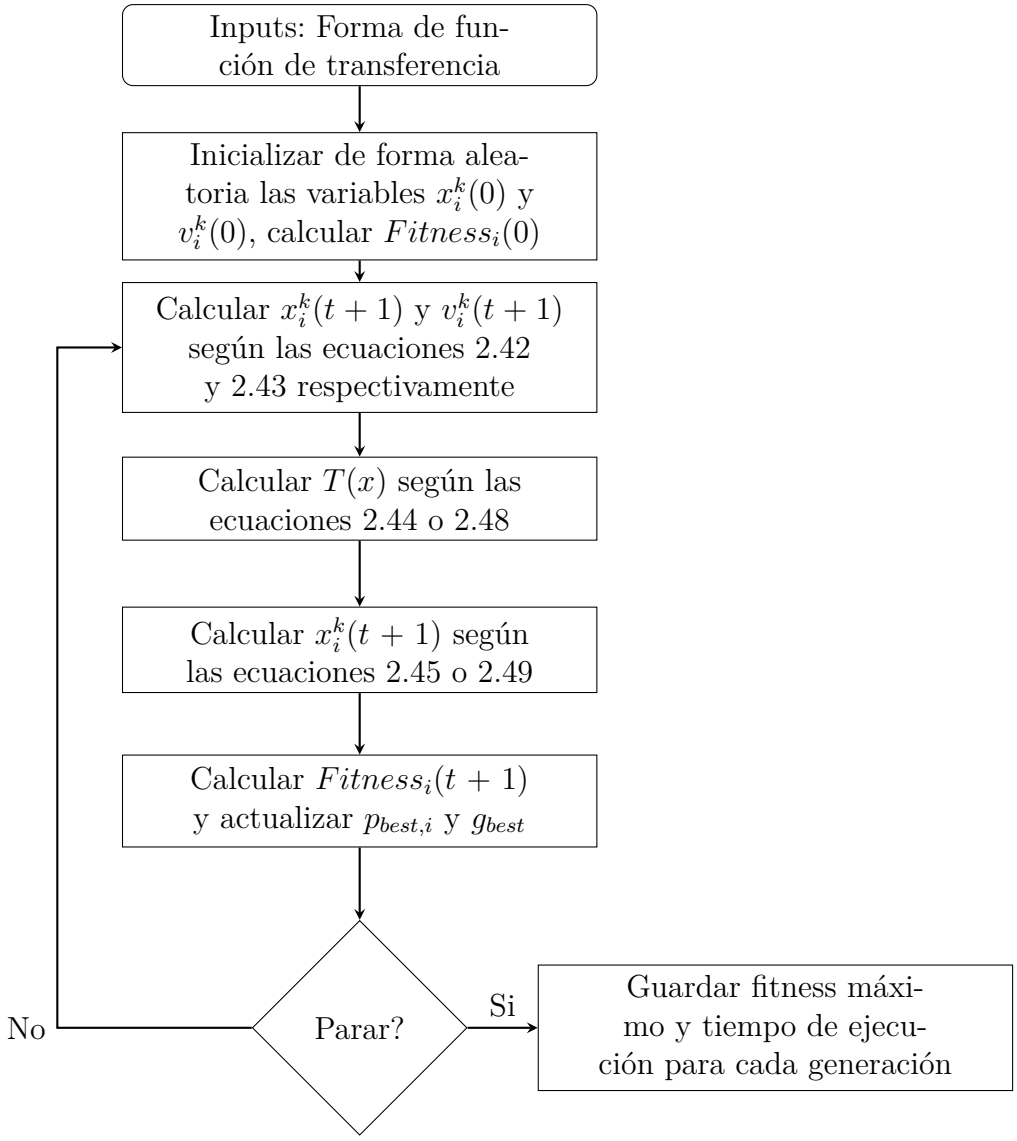


Figura 2.5: Diagrama del algoritmo MBPSO.

Capítulo 3

Metodología

La metodología de este trabajo de tesis consiste en la implementación de dos algoritmos de optimización (GA y MBPSO), los cuales son aplicados a placas 2D representadas mediante dos algoritmos o modelos diferentes (IGA y FEA). Las principales etapas para la realización de este trabajo son la siguientes.

- Búsqueda bibliográfica sobre métodos para el modelado de placas y la optimización de bandgaps, específicamente se busca sobre IGA, FEA, GA, PSO, BPSO, MBPSO.
- Familiarización con el software Comsol Multiphysics, con el cual se aprende a modelar diferentes tipos de placas, aplicando las condiciones de borde correspondientes, con el fin de determinar diagramas de bandas.
- Programar los métodos numéricos asociado a FEA e IGA para modelar placas 2D, los algoritmos de optimización GA y MBPSO para optimizar los bandgaps. FEA se programa utilizando el Toolbox SDTools de Matlab.
- Validar los modelos FEA e IGA, comparando frecuencias naturales para placas completamente libres con bibliografía pertinente y también usando los resultados arrojados por Comsol Multiphysics.
- Validar correcta implementación de condiciones de borde periódicas, comparando diagramas de bandas obtenidos con los arrojados por Comsol Multiphysics.
- Optimización de bandgaps para todas las combinaciones de modelos y algoritmos de optimización, utilizando diferentes números de elementos por malla. La optimización se realiza 10 veces con el fin de posteriormente comparar promedios y desviaciones estándar.
- Determinar y comparar diagramas de convergencia, topologías y diagrama de bandas para cada caso.
- Realizar optimización topológica y optimización del radio de orificio para placas cuadradas con orificios circulares, utilizando la combinación mas adecuada entre los algoritmos usados.

A continuación se presenta un diagrama donde se ilustra la metodología utilizada en esta investigación.

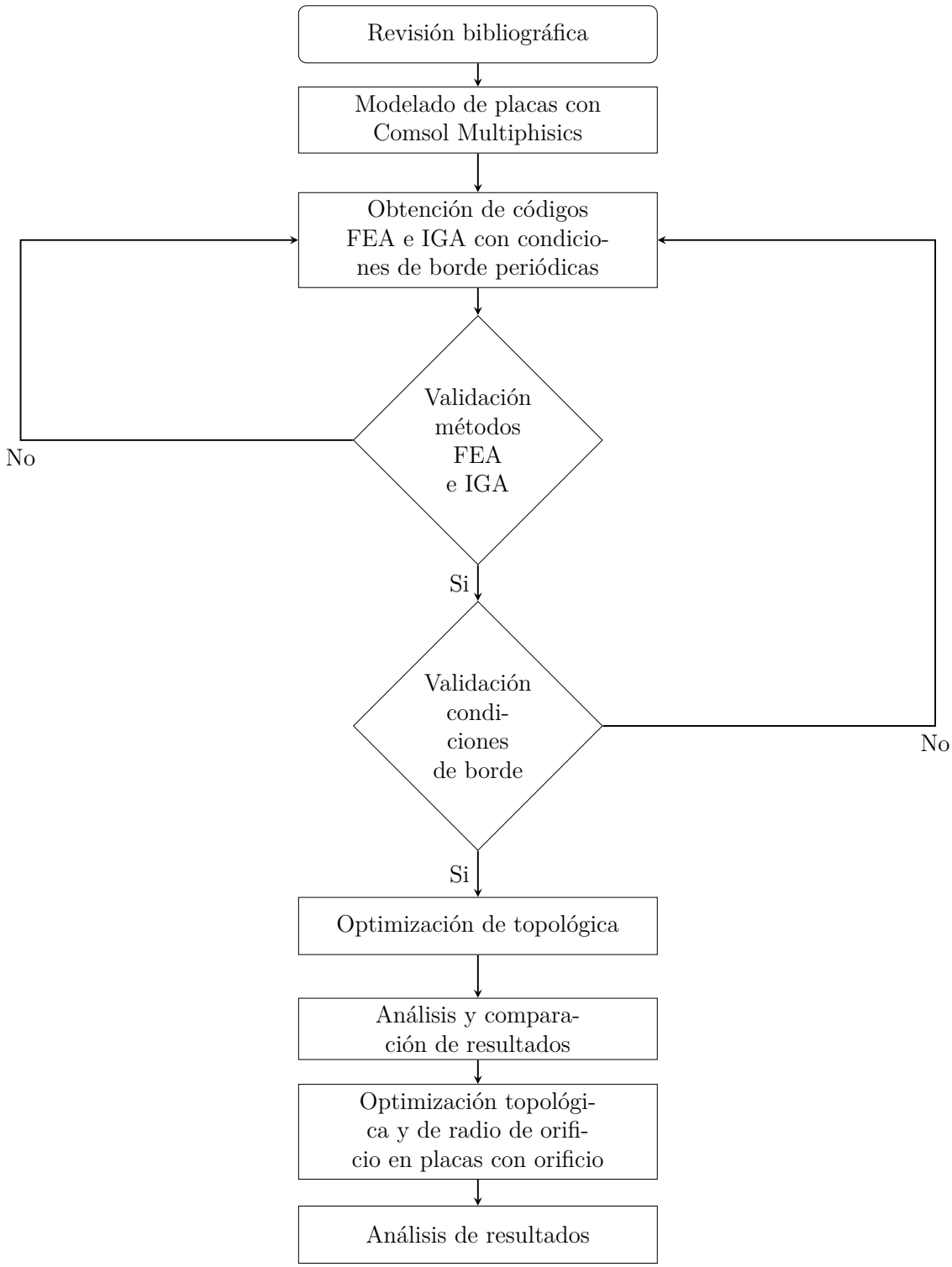


Figura 3.1: Diagrama de metodología usada en esta investigación.

Recursos

Los recursos utilizados en este trabajo están relacionados a los sofware empleados en este trabajo. Para todos los códigos se utiliza el software Matlab en su versión 2018b, disponible en Microsoft Windows, MAC OS X y diferentes distribuciones de sistemas operativos basados en Linux.

En cuanto al hardware utilizado, se programa, validan y obtienen resultados en un notebook MSI con una CPU Inter Core i7-9750H a una frecuencia base de 2.60GHz, con 6 núcleos y 12 procesadores lógicos, 32.0 GB de memoria RAM, sistema operativo Windows 10 Pro, 64 bits.

3.1. Función objetivo

Para determinar la función objetivo del problema se debe tener en cuenta que existen dos tipos de topologías que pueden resultar de la optimización, las que son factibles y las que no. Las primeras corresponden a las placas cuya topolgía es tal que todos los elementos que la conforman están conectados por al menos una arista, mientras que la segunda son las que tiene al menos un elemento no conectado con el resto o tienen elementos conectados por un vértice.

Dentro de estos dos tipos de placas, está la posibilidad que existan vacíos del tamaño mínimo posible según la discretización de la placa, cuando esto ocurre, se procede a reparar o rellenar estos vacíos, con el fin de facilitar la posible futura manufactura de la placa, considerando que el elemento mínimo que constituye una placa es muy pequeño en comparación a las dimensiones de esta. También se cumple que los diagramas de bandas de la placa antes y después de la reparación no sufren mayores modificaciones.

En base a lo anterior se debe definir una función objetivo que tenga en cuenta esta diferencia, para esto se debe considerar primero una función objetivo $f(x)$ definida como la resta de los valores de las frecuencias asociados a los bordes de dos bandas contiguas, o sea una función que calcule el bandgap en una placa, es decir:

$$f(x) = min(w_{i+1}) - max(w_i) \tag{3.1}$$

Donde i es el número de una banda en particular, luego $min(w_{i+1})$ y $max(w_i)$ corresponden a los valores máximos y mínimos de dos bandas contiguas, x corresponde a un vector binario que representa la topología de la placa.

Para modificar la función objetivo se deben tener en cuenta los siguiente criterios:

- La reparación de la placa (en el sentido antes mencionado) se realiza previo a la evaluación de la función objetivo.
- Una placa con una topología factible es siempre preferible por sobre una infactible.
- Al comparar dos topologías factibles es preferible una cuya función objetivo sea mayor.
- Al compara dos topologías infactibles es preferible una cuya función objetivo sea mayor.

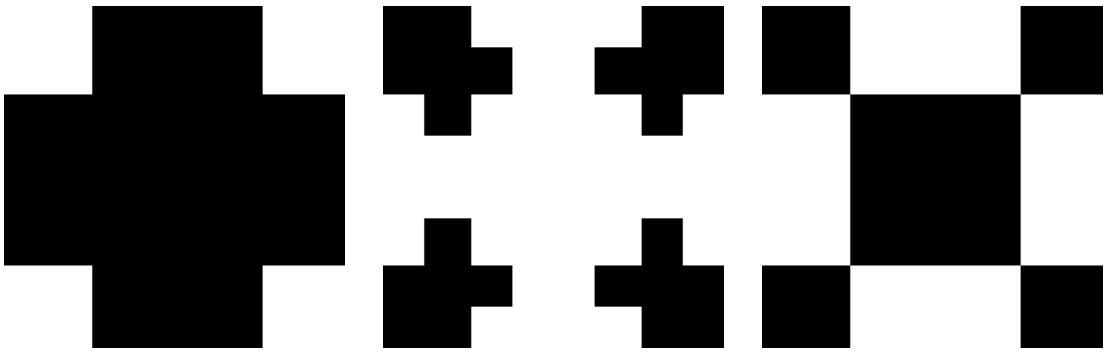
Teniendo en cuenta los criterios anteriores, la función objetivo se modifica de la siguiente manera:

$$f'(x) = f(x) - Cfact(x) \tag{3.2}$$

Donde C es una contante real de un valor mayor al mayor valor posible de $f(x)$ cuando x representa una geometría infactible, $fact(x)$ es una función binaria que toma el vector binario e identifica si es factible o no, atribuyendo el valor 1 si la geometría es infactible y un cero en el caso contrario, es decir:

$$fact(x) = \begin{cases} 0 \quad si \quad \text{x representa geometría factible} \\ \\ 1 \quad si \quad \text{x representa geometría infactible} \end{cases} \tag{3.3}$$

A continuación se presentan ejemplos de las diferentes posibilidad de placas, donde la placa de la figura 3.2a es una factible, la placa de la figura 3.2b es infactible dado que no están todos sus elementos unidos entre si y la placa de la figura 3.2c es infactible porque existen elementos unidos por vértices. El color negro representa la placa, mientras que el color blanco a vacíos de la misma.



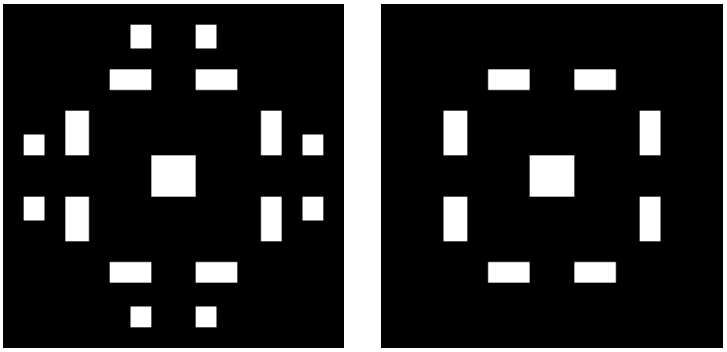
(a) Placa factible.

(b) Placa infactible por no poseer todos sus elementos conectados.

(c) Placa infactible por poseer elementos conectador por algún vértice.

Figura 3.2: Ejemplos de placas factible e infactibles.

En la figura 3.3a se muestra un ejemplo de una placa que necesita ser reparada, mientras en la figura 3.3b se muestra la misma placa luego de realizarse la reparación, cabe destacar que los únicos vacíos reparados son los del tamaño mínimo posible según la discretización de la placa, luego no todos los vacíos deben ser reparadaos. Para este ejemplo se usan placas de 16 elementos por lado.



(a) Placa a reparar. (b) Placa luego de ser reparada.

Figura 3.3: Ejemplos de placa a reparar y placa reparada.

3.2. IGA

Para determinar de forma satisfactoria el diagrama de bandas de una placa usando el modelo IGA se usa una función general que unifica todas las funciones necesarias para la creación de la placa, cálculo de frecuencias y modos normales y cálculo del bandap, la función se resume como sigue.

Algoritmo 1: Función distIGA

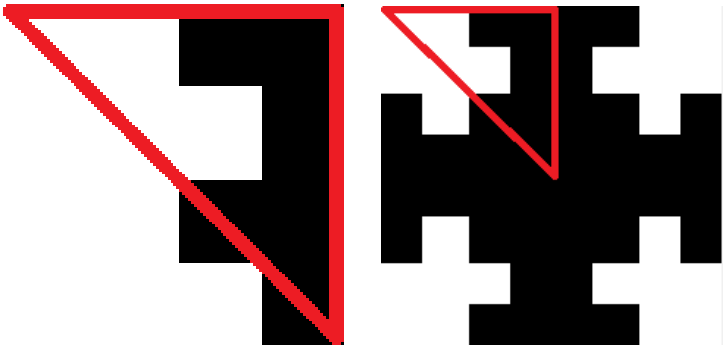
Input : Vector binario **x** antes de aplicarse condiciones de simetría del problema, número de la banda n_{banda} para calcular la diferencia de frecuencia y número de modos n_{modos} a calcular en cada banda

Output: dist: Distancia entre banda n_{banda} y la banda $n_{banda} + 1$

inicializacion;
X = simetria(x);
[K,M] = placaIGA(X);
[ω] = frecuenciasIGA(K,M,n_{modos});
dist = distancia(ω,X,n_{banda});

Para describir la función llamada simetría es conveniente explicarla con ejemplos gráficos en vez de con una descripción general del código.

Se parte de un vector binario **x**, el cual representa una sección triangular u octante de la placa, por ejemplo si se tiene que $\mathbf{x} = [0, 0, 1, 1, 0, 0, 1, 1, 1, 1]$, este vector representa la sección triangular de la figura 3.4a, mientras que al realizar las operaciones de simetría se obtiene la placa de la figura 3.4b. El número uno del vector binario indica que hay material, lo cual es representado con el color negro, mientras que el número cero indica que no hay material, lo cual se representa con el color blanco.



(a) Sección triangular antes de operaciones de simetría.

(b) Placa luego de operaciones de simetría.

Figura 3.4: Dos tipos de diagramas de bandas obtenibles.

Las operaciones de simetría antes mencionadas son 3, la primera consiste en una reflexión de la sección triangular por su hipotenusa, de tal forma de lograr un cuadrado que forma un cuadrante de la placa, luego este cuadrado se refleja de su arista derecha, resultando un rectángulo, luego este rectángulo se refleja por su arista inferior, logrando finalmente la placa completa.

Para la generación de las matrices de masa y rigidez que representan la placa se parte del vector binario luego de aplicarse las condiciones de simetría del problema, con los cuales se generan los puntos de control y knots que definen la placa en el espacio de parámetros según IGA (algoritmo 2).

Según los algoritmos 3 y 8, encargados del cálculo de las frecuencias del sistema, se toman las matrices de masa y rigidez previamente calculadas y se les aplican las condiciones de borde periódicas del sistema, definidas por una matriz de transformación $\mathbf{T}$. Se realiza el cálculo de un número n_{modos} de valores propios para cada valor del vector de onda en el recorrido definido en la primera zona irreductible de Brillouin.

El cálculo del bandgap se realiza con el algoritmo 4, el cual toma las frecuencias calculadas con IGA o FEA, el número del modo normal donde se realiza el cálculo de la banda y el vector binario luego de calcular las condiciones de simetría del problema, con esta información se calcula la función objetivo original del problema, luego un factor binario que indica si la placa es factible o no, para luego calcular la función objetivo final.

Algoritmo 2: Función placaIGA

Input : Vector binario $\mathbf{X}$ luego de aplicar las condiciones de simetría
Output: Matrices de masa y rigidez de la placa según IGA
Inicializa parámetros que definen las propiedades mecánicas del material que compone la placa y las dimensiones de la misma: E, ν, ρ, a, h, $D = \frac{E}{1-\nu^2}$, los cuales son el módulo de elasticidad, módulo de Poisson, densidad, largo de la arista de la placa, espesor de la placa y la rigidez a la flexión respectivamente;
Define knots ξ_i y η_j y puntos de control $\mathbf{T}_{i,j}$ según el tamaño de $\mathbf{X}$;
Determina número de elementos totales en la placa **noElems**;
Prelocaliza matrices de rigidez y masa $\mathbf{K}$ y $\mathbf{M}$;
Determina matrices $\mathbf{Q}$ y $\mathbf{W}$ con puntos de cuadratura gaussiana de la placa;
for *1 ⟵ e to* ***noElems*** **do**
  for *1 ⟵ gp to size(* $\boldsymbol{W}$*)* **do**
    Determina ξ_i y η_j pertenecientes al knot span;
    Calcula $N_{i,p}(\xi)$ y $M_{j,q}(\eta)$ según las ecuaciónes A.9 y A.10 ;
    Calcula $R^{p,q}_{i,j}(\xi,\eta)$ y cada una de sus derivadas con respecto a cada coordenada paramétrica según la ecuación A.12;
    Con las derivadas de $R^{p,q}_{i,j}(\xi,\eta)$ según las ecuaciones A.13 a la A.15, donde las derivadas respecto η son análogas ;
    Calcula $\mathbf{B}_I$ según la ecuación 2.22 ;
    Calcula $\mathbf{M}_{I,J}$ y $\mathbf{K}_{I,J}$ según las ecuaciónes 2.20 y 2.21 ;
  end
end

Algoritmo 3: Función frecuenciasIGA

Input : Matices de masa y rigidez $\mathbf{M}$ y $\mathbf{K}$, número de modos a calcular n_{modos}.
Output: Matrices con frecuencias ω.
Crear vector k con valores del vector de onda pertenencientes a la primera zona irreductuble de Brillouin;
Inicializar matrices $\mathbf{T_1}$, $\mathbf{T_2}$, $\mathbf{T}$ y ω;
for *1 ⟵ i to* ***size(*** k***)*** **do**
  Cálculo de matrices $\mathbf{T_1}(\mathbf{k_i})$ y $\mathbf{T_2}(\mathbf{k_i})$ según las ecuaciones 2.34 y 2.35 ;
  Cálculo de matriz $\mathbf{T}(\mathbf{k_i})$ con las matrices antes calculadas, según la ecuación 2.36 ;
  Calcular matriz $\widetilde{\mathbf{D}}(k_i) = \mathbf{T}^{\dagger}(\mathbf{K} - \omega^2\mathbf{M})\mathbf{T}$;
  Cálculo de n_{modos} frecuencias normales $\omega(k_i)$ utilizando $\widetilde{\mathbf{D}}(k_i)$;
end

Algoritmo 4: Función distancia

Input : Matriz de frecuencias ω y número de la banda objetivo n_{banda}, vector binario $\mathbf{X}$, calculado luego de aplicar las condiciones de simetría del problema
Output: distancia dist entre dos bandas contiguas según la función objetivo $f'(X)$
Calculo de función binaria fact(X) según la ecuación 3.1;
Calculo del valor de la función objetivo original $f(x)$ según la ecuación 3.3;
Calculo de la función objetivo $f'(X)$ luego de la modificación, según la ecuación 3.2;

Para determinar diagramas de bandas en placas con orificio se deben hacer ciertas modi-

ficaciones la función que determinar las matrices de masa y rigidez del sistema, para esto se crea una nueva función llamada placaConOrificioIGA, la cual es creada a partir de la función placaIGA (algoritmo 2), tiene el fin el fin de modelar una placa con orificio de tal forma que se cree un orificio de radio r_c en el centro de la misma, también se identifican los elementos de la placa que están completa y parcialmente dentro del orificio, con el fin de fijar los grados de libertad correspondientes. Es importante señalar que al fijar los grados de libertad antes mencionados, estos se eliminar del del vector binario que define la placa, disminuyendo así la disensión del espacio de optimización.

Algoritmo 5: Función placaConOrificioIGA

Input : Vector binario $\mathbf{X}$ luego de aplicar las condiciones de simetría y radio de orificio r_c

Output: Matrices de masa y rigidez de la placa con orificio de radio r_c.

Inicializa parámetros que definen las propiedades mecánicas del material que compone la placa y las dimensiones de la misma: E, ν, ρ, a, h, $D = \frac{E}{1-\nu^2}$, los cuales son el módulo de elasticidad, módulo de Poisson, densidad, largo de la arista de la placa, espesor de la placa y la rigidez a la flexión respectivamente;

Define knots ξ_i y η_j y puntos de control $\mathbf{T}_{i,j}$ según el tamaño de $\mathbf{X}$;

Determina número de elementos totales en la placa **noElems**;

Prelocaliza matrices de rigidez y masa $\mathbf{K}$ y $\mathbf{M}$;

Determina elementos completamente dentro del orificio **inactiveElems**;

Determina elementos parcialmente dentro del orificio **splitElems**;

for *1 ⟵ e to* ***noElems*** **do**

  if *e ∈****inactiveElems*** **then**

    continue;

  end

  if *e ∈****splitElems*** **then**

    Determina matrices $\boldsymbol{Q}$ *y* $\boldsymbol{W}$ *con puntos de cuadratura gaussiana al rededor del orificio;*

  end

  else

    Determina matrices $\boldsymbol{Q}$ *y* $\boldsymbol{W}$ *con puntos de cuadratura gaussiana fuera del orificio;*

  end

  for *1 ⟵ gp to size(*$\boldsymbol{W}$*)* **do**

    Determina ξ_i *y* η_j *pertenecientes al knot span;*

    Calcula $N_{i,p}(\xi)$ *y* $M_{j,q}(\eta)$ *según las ecuaciónes A.9 y A.10 ;*

    Calcula $R^{p,q}_{i,j}(\xi,\eta)$ *y cada una de sus derivadas con respecto a cada coordenada paramétrica según la ecuación A.12;*

    Con las derivadas de $R^{p,q}_{i,j}(\xi,\eta)$ *según las ecuaciones A.13 a la A.15, donde las derivadas respecto* η *son análogas ;*

    Calcula $\mathbf{B}_I$ *según la ecuación 2.22 ;*

    Calcula $\mathbf{M}_{I,J}$ *y* $\mathbf{K}_{I,J}$ *según las ecuaciónes 2.20 y 2.21 ;*

  end

end

La única diferencia entre el algoritmo 5 y el algoritmo original (algotirmo 2) radica en

que se determina una cuadratura gaussina diferente alrededor del orificio, con el fin de que la integración sea mas precisa en ese borde, aumentando el numero de puntos de integración.

La figura 3.5 muestra de forma visual como se dividen los elementos de la placa, cada elemento se clasifica y agrupa en cientos grupos definidos luego de la creación del círculo.

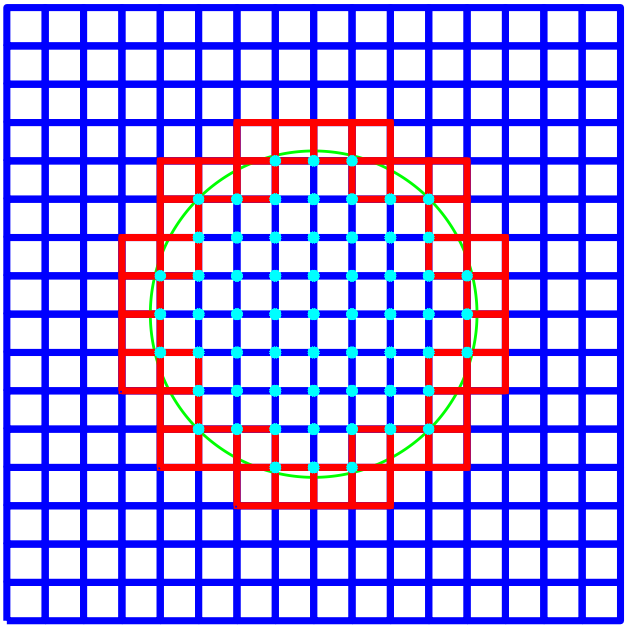


Figura 3.5: Ejemplo de placa de 16x16 elementos con sus elementos diferenciados según corresponda para un radio de orificio de 0.026 [m].

Donde el círculo de color verde representa el límite del orificio, las lineas marcadas de color azul, encerradas por el orificio, corresponden a las aristas que conforman cada elemento que se procede a eliminar de la placa, los puntos de color cyan corresponden a los nodos de los elementos anteriormente mencionados, las lineas marcadas de color rojo corresponden a las aristas de los elementos que bordean el orificio y tienen segmentos parcialmente dentro de este, estos elementos se fijar de tal forma que siempre tienen que ser construidos. Por último, las lineas de color azul al exterior del círculo, representan a los elementos que no tienen ningún segmento dentro del orificio, y por lo tanto son los elementos a los cuales se les realiza la optimización topológica, definiendo si deben ser construidos o no.

Dado que los elementos que están parcial y completamente dentro del círculo se fijan de tal forma que son forzados a construirse y no construirse respectivamente, se disminuye el número de elementos por definir en la optimización. Se cumple que a mayor radio, mayor es el número de elementos que son fijados.

3.3. FEA

Para determinar los los diagramas de bandas de una placa usando el método FEA, al igual que para el caso del método IGA, se usa una función general que unifica el set de funciones utilizado para la creación de la placa, cálculo de frecuencias y modos normales y cálculo de bandap, se función se resume de la siguiente manera:

Algoritmo 6: Función distFEA

Input : Vector binario **x** antes de aplicarse condiciones de simetría del problema, número de la banda n_{banda} para calcular la diferencia de frecuencia y número de modos n_{modos} a calcular en cada banda

Output: dist: Distancia entre banda n_{banda} y la banda $n_{banda} + 1$

X = simetria(x);
[K,M] = placaFEA(X);
[ω] = frecuenciasFEA(K,M,n_{modos});
dist = distancia(ω,X,n_{banda});

La función anterior difiere de disIGA solo en las funciones destinadas a calcular las matrices de masa y rigidez y en el calculo de frecuencias y modos normales, luego las funciones simetría y distancia son las mismas, por lo que no se vuelve a explicarlas.

Dado que se usa una Toolbox de elementos finitos en Matlab la función placaFEA es mucho mas simple y menos detallada en comparación a su contraparte en IGA. Cabe destacar que los nodos que no aparecerán finalmente en la placa no son considerados en la construcción de la misma, luego el tamaño de las matrices de masa y rigidez de una placa es variable, y depende de la cantidad de nodos que posea cada una.

Para la generación de las matrices de masa y rigidez que representan la placa se parte de un vector binario luego de aplicarse las condiciones de simetría del problema, con el que se genera un arreglo de puntos, con los cuales se crean los elementos que definen la placa cuadrada, los que son entregados como argumento a una función que crea las matrices de masa y rigidez de forma automática (algoritmo 7).

El algoritmo 7 es considerablemente mas corto y simple en comparación a su contraparte en IGA porque para el caso de FEA se trabaja con una Toolbox de elementos finitos, luego hay muchas funciones y procedimientos que están automatizados y optimizados, por lo que no hay necesidad de realizar nuevos códigos, a diferencia del caso de IGA, donde los procedimientos no están automatizados y deben de ser programados desde cero.

Algoritmo 7: Función placaFEA

Input : Vector binario $\mathbf{X}$ luego de aplicar las condiciones de simetría
Output: Matrices de masa y rigidez de la placa según FEA
Inicializa parámetros que definen las propiedades mecánicas del material que compone la placa y las dimensiones de la misma: E, ν, ρ, a, h, $D = \frac{E}{1-\nu^2}$, los cuales son el módulo de elasticidad, módulo de Poisson, densidad, largo de la arista de la placa, espesor de la placa y la rigidez a la flexión respectivamente;
A parir de $\mathbf{X}$ se crea un arreglo con los nodos que estarán finalmente en la placa y otro con los elementos de la misma;
Fija dos los dos grados de libertad perpendiculares a la normal de la placa y un grado de libertad rotacional;
Con la información de los nodos, elementos y los grados de libertad, se crean las matrices de masa $\mathbf{M}$ y rigidez $\mathbf{K}$ de la placa;

Algoritmo 8: Función frecuenciasFEA

Input : Matices de masa y rigidez $\mathbf{M}$ y $\mathbf{K}$, número de modos a calcular n_{modos}
Output: Matrices con frecuencias ω.
Crear vector k con valores del vector de onda pertenencientes a la primera zona irreductuble de Brillouin;
Inicializar matrices $\mathbf{T_1}$, $\mathbf{T_2}$, $\mathbf{T}$ y ω;
for *1* $\longleftarrow$ *i to* ***size(****k****)*** **do**
Cálculo de matrices $\mathbf{T_1}(k_i)$ *y* $\mathbf{T_2}(k_i)$ *según las ecuaciones 2.26 y 2.27;*
Cálculo de matriz $\mathbf{T}(k_i)$ *con las matrices antes calculadas, según la ecuación 2.28;*
Se identifican las filas y columnas eliminadas de las matrices $\mathbf{M}$ *y* $\mathbf{K}$*, para posteriormente eliminarlas también de* $\mathbf{T}$*;*
Calcular matriz $\widetilde{\mathbf{D}}(k_i) = \mathbf{T}^{\dagger}(\mathbf{K} - \omega^2\mathbf{M})\mathbf{T}$*;*
Cálculo de n_{modos} *frecuencias normales* $\omega(k_i)$ *utilizando* $\widetilde{\mathbf{D}}(k_i)$*;*
end

3.4. MpGA

Para realizar la optimización utilizando algoritmos genéticos se parte con una función que inicialice las 4 poblaciones a utilizar, la cual se muestra a continuación.

Algoritmo 9: Función mainGA

Input : Número n de elementos por arista de la placa a optimizar, valor n_{banda} de la frecuencia normal para el cálculo de la banda.
Output: Mejor individuo, valor del fitness, tiempo de ejecución y curva de convergencia para cada población
Calcular número de genes n_{genes} para iniciar la optimización;
Inicializar hiperparámetros a usar en los algoritmos genéticos: tamaño de población p_{size}, probabilidad de entrecruzamiento p_c, probabilidad de mutación p_m y número de generaciones n_g para comprobar criterio de convergencia;
Iniciar función allGA;

Para cada una de las 4 poblaciones generadas se tienen dos funciones, la primera, llamada allGA, ejecuta algoritmos genéticos de forma sucesiva hasta alcanzar el criterio de convergencia global de esta población, criterio mostrado en la ecuación 2.41, también de realizar la migración de los mejores individuos de cada población a las demás, reemplazándolos por los peores, también se encarga de juntar toda la información referente a la optimización entregada por cada población. La segunda función, llamada GA, es llamada de forma sucesiva por el primer algoritmo, siguiendo el criterio de convergencia mostrado en la ecuación B.8, se encaga de la evolución de cada población, generando la población de hijos, realizar entrecruzamiento y mutación y guardar la información de la optimización.

Algoritmo 10: Función allGA

Input : Modelo de placa a utilizar, n_{genes}, t_p, p_c, p_m y n_g
Output: Información de la optimización
Crear población inicial de forma aleatoria;
$0 \longleftarrow \mathbf{conv_1}$;
while $\sim \mathbf{conv_1}$ **do**
  for *1* $\longleftarrow$ *i to* t_p **do**
    Ejecuta la función reparar;
    Calcula el valor de su función objetivo
  end
  Ejecuta la función GA;
  Carga el valor del mejor individuo de las otras poblaciones;
  Reemplaza mejores individuos de las otras poblaciones con los perores de la población actual;
  Evalúa criterio de convergencia según la ecuación 2.41 y guarda su valor en $\mathbf{conv_1}$*;*
  Guarda información de la optimización;
end

Algoritmo 11: Función GA

Input : Población inicial, n_{genes}, p_{size}, p_c, p_m y n_g
Output: Población final, información de la optimización
$0 \longleftarrow \mathbf{conv_2}$;
while $\sim \mathbf{conv_2}$ **do**
  Guarda mejor individuo;
  Genera población de hijos;
  Realiza entrecuruzamiento;
  Realiza mutación;
  Ejecuta la función reparar;
  Calcula valor de la función objetivo ;
  Evalúa criterio de convergencia de la ecuación B.8 y guarda su valor en $\mathbf{conv_2}$;
  Guarda información de optimización;
  Actualizar población de hijos como nueva población inicial;
end

3.5. MBPSO

Para realizar la optimización utilizando el método MBPSO se parte, al igual que su contraparte de MpGA, se encarga de iniciar los hiperparámetros de la optimización y que llama o usa el optimizador en si.

El algoritmo 13 es el encargado de realizar la optimización según el método MBPSO, en el cual se calcula la función objetivo, calcular posiciones y velocidades de cada partícula, evaluar el criterio de convergencia y juntar toda la información de la optimización.

Algoritmo 12: Función mainMBPSO

Input : Número n de elementos por arista de la placa a optimizar, valor n_{banda} de la frecuencia normal para el cálculo de la banda.

Output: Mejor individuo, valor del fitness, tiempo de ejecución y curva de convergencia

inicializacion;
Calcular número de genes o variables n_g para iniciar la optimización;
Inicializar hiperparámetro número de partículas n_p;
Calcula número de variables n_{genes} del modelo (análogo al número de genes en GA);
Inicializar optimizador;

Algoritmo 13: Función MBPSO

Input : Número n de elementos por arista de la placa a optimizar, número de partículas n_p,número de variables del modelo n_{genes}

Output: Mejor individuo, valor del fitness, tiempo de ejecución y curva de convergencia

Inicializa valores de hiperparámetros adicionales: velocidad máxima v_{max} y constantes C_1 y C_2;

Genera n_p partículas iniciales de forma aleatoria;

$0 \longleftarrow \mathbf{conv_3}$;

while $\sim \mathbf{conv_3}$ **do**

  for *1* $\longleftarrow$ *i to* n_p **do**

    Ejecuta la función reparar;

    Calcula valor de la función objetivo;

    Determina g_{best} *y* p_{best}*;*

    for *1* $\longleftarrow$ *j to* n_{genes} **do**

      Calcula la velocidad $v_{i,j}$ *según la ecuación 2.42 ;*

      Evalúa la condición para la velocidad máxima o mínima del problema, como se muestra en la ecuación B.2;

      Calcula el valor de la función de transferencia $T(v_{i,j})$ *según la cuarta expresión de la ecuación 2.48;*

      Actualiza valor de la componente para la nueva población de partículas según la ecuación 2.49 ;

    end

    Evalúa criterio de convergencia de la ecuación B.8 y guarda su valor en $\mathbf{conv_3}$*;*

    Guarda información de la optimización;

  end

end

Capítulo 4

Resultados

4.1. Validación de algoritmos

A continuación se presentan los resultados de la validación de los modelos usados y de la correcta implementación de las condiciones de borde, se contrastan los resultados con los obtenidos usando el software comercial Comsol Multiphysics, la validación se divide en 3 partes, donde cada una se realiza para IGA y FEA, estas son: validación de modelos calculando primeros modos de vibración en placa cuadrada completamente libre, validación de condiciones de borde periódicas para una placa cuadrada, validación de condiciones de borde y modelo para placa con un sacado en cada esquina, como se puede ver en las figuras 4.2a y 4.2b.

En la tabla 4.1 y 4.2 se muestran las frecuencias asociadas a los primeros 8 modos normales de una placa cuadrada de 10[m] de longitud de arista, modeladas con FEA e IGA respectivamente. Se varía la cantidad desde 8 hasta 128 la cantidad de elementos por arista. La tabla 4.3 muestra las frecuencias asociadas a la placa modelada con Comsol, en la cual se usa un mallado *Normal*, el cual corresponde al mallado que por defecto es aplicado a una superficie, el que también se ajusta automáticamente a la forma y tipo de superficie usada.

Tabla 4.1: Valores de frecuencias obtenidos en FEA para los primeros 8 modos normales de una placa libre, para diferente cantidad de elementos por placa.

Modo	8x8	16x16	32x32	64x64	128x128
w_1 [Hz]	0.0000	0.0000	0.0000	0.0000	0.0001
w_2 [Hz]	0.0000	0.0000	0.0000	0.0000	0.0001
w_3 [Hz]	0.0000	0.0000	0.0000	0.0000	0.0001
w_4 [Hz]	1.6230	1.6220	1.6217	1.6217	1.6217
w_5 [Hz]	2.3581	2.3591	2.3594	2.3595	2.3595
w_6 [Hz]	2.9196	2.9215	2.9221	2.9222	2.9223
w_7 [Hz]	4.1977	4.1918	4.1906	4.1903	4.1903
w_8 [Hz]	4.1977	4.1918	4.1906	4.1903	4.1903

Tabla 4.2: Valores de frecuencias obtenidos en IGA para los primeros 8 modos normales de una placa libre, para diferente cantidad de elementos por placa.

Modo	8x8	16x16	32x32	64x64	128x128
w_1 [Hz]	0.0000	0.0000	0.0000	0.0000	0.0000
w_2 [Hz]	0.0000	0.0000	0.0000	0.0000	0.0000
w_3 [Hz]	0.0000	0.0000	0.0000	0.0000	0.0000
w_4 [Hz]	1.6217	1.6216	1.6216	1.6216	1.6216
w_5 [Hz]	2.3596	2.3594	2.3594	2.3594	2.3594
w_6 [Hz]	2.9225	2.9222	2.9221	2.9221	2.9221
w_7 [Hz]	4.1909	4.1900	4.1900	4.1900	4.1900
w_8 [Hz]	4.1909	4.1900	4.1900	4.1900	4.1900

Tabla 4.3: Valores de frecuencias obtenidos en Comsol para los primeros 8 modos normales de una placa libre.

Modo	Comsol
w_1 [Hz]	0.00
w_2 [Hz]	0.00
w_3 [Hz]	0.00
w_4 [Hz]	1.62
w_5 [Hz]	2.35
w_6 [Hz]	2.91
w_7 [Hz]	4.18
w_8 [Hz]	4.18

La figura 4.1 muestra el gráfico de la suma de los errores de las 8 frecuencias naturales calculadas con FEA e IGA al ser comparadas con las entregadas por Comsol, se tienen diferentes valores en función del número de elementos usados para el modelado de la placa.

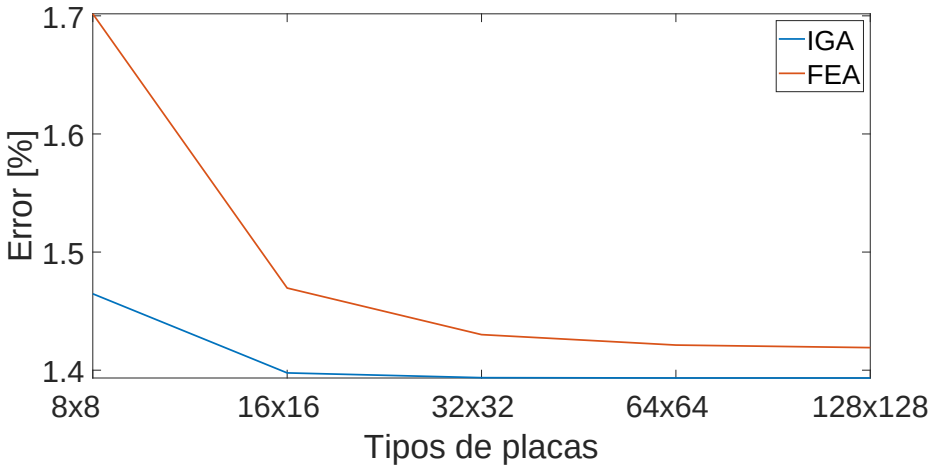


Figura 4.1: Gráfico de error porcentual de frecuencias calculadas con FEA e IGA al ser comparadas con Comsol.

Según la figura 4.1, se tiene que el error asociado al cálculo de frecuencias normales decrece conforme aumenta el número de elementos que posea una placa, también se cumple que el valor del error se estabiliza en un valor en específico (diferente en cada caso), cuando esto sucede se cumple que los valores entregados son independientes del mallado.

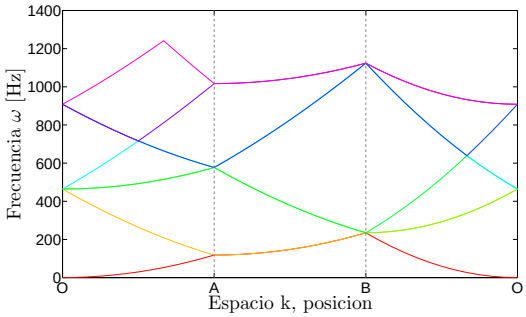
(a) Placa completa

(b) Placa con sacado en sus cuatro esquinas.

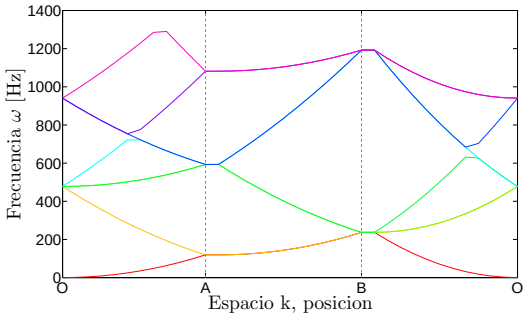
Figura 4.2: Dos tipos de placas usadas para la obtención de diagramas de bandas durante la validación.

Las figuras 4.3 y 4.4 muestran los diagramas de bandas para una placa completa y una placa con sacados en sus cuatro esquinas (ver figura 4.2), modeladas con Comsol, FEA e IGA. De los gráficos se tienen diferencias mínimas en cada caso, dado que en cada uno los valores de las frecuencia siguen los mismos comportamientos y se mantienen en los mismos rangos. Solo se observan diferencias menores en ciertos puntos de los diagramas, donde las frecuencias tienen comportamientos casi constantes cuando se juntan dos o mas mondos normales, por ejemplo cuando se juntan los modos 3, 4, 5 y 6 en el punto **A** o cuando se juntan los modos 1, 2, 3 y 4 en el punto **B**, lo anterior sucede para ambas figuras.

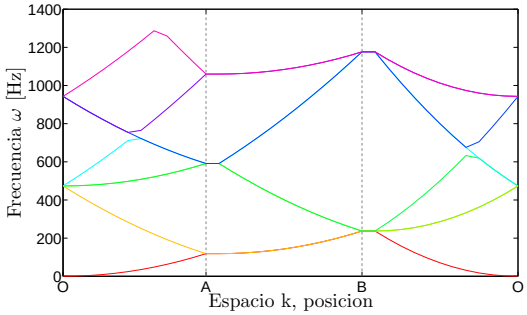
Dado que los algoritmos utilizados en la creación y cálculo de frecuencias normales en placas con condiciones de borde periódicas tienen resultados muy similares a los arrojados por Comsol Multiphysics y se deja de depender de la malla para placas de entre 16 y 32 elementos por lado, se considera que estos algoritmos son validados y pueden ser usados para realizar las optimizaciones futuras.



(a) Diagrama de Bandas obtenido con Comsol.

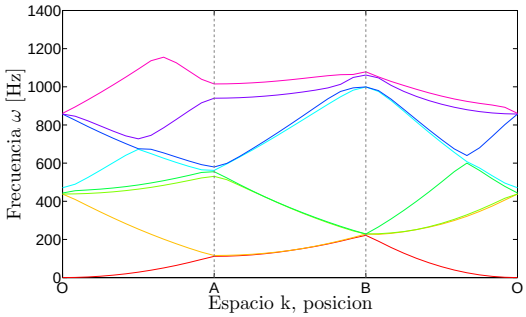


(b) Diagrama de Bandas obtenido con FEA.

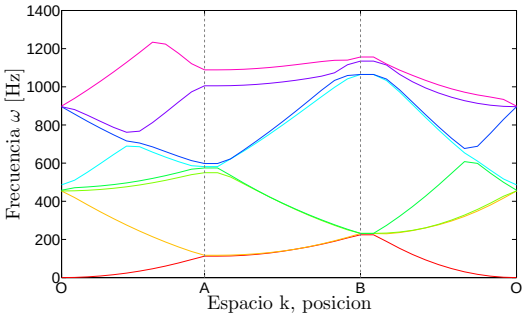


(c) Diagrama de Bandas obtenido con IGA.

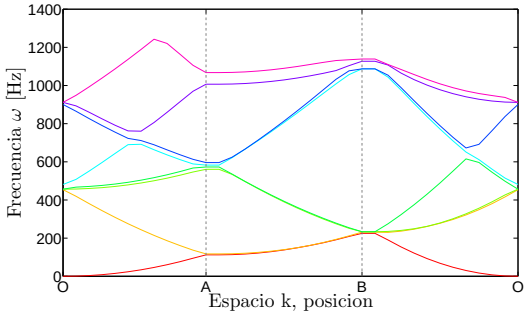
Figura 4.3: Diagramas de Bandas para placa completa.



(a) Diagrama de Bandas obtenido con Comsol.



(b) Diagrama de Bandas obtenido con FEA.



(c) Diagrama de Bandas obtenido con IGA.

Figura 4.4: Diagramas de Bandas para placa con sacado en sus cuatro esquinas.

4.2. Optimización topológica

4.2.1. Parámetros usados en el modelado y optimización

En la tabla 4.4 se presentan los parámetros físicos usados para el modelado de cada una de las placas.

Tabla 4.4: Parámetros físicos placas para ambos modelos usados

Parámetro	Valor
a	1 [m]
b	1 [m]
h	0.05 [m]
E	200 [GPa]
ν	0.3
ρ	8000 $[\frac{Kg}{m^3}]$

donde a y b son las dimensiones horizonal y vertical respectivamente, h es el espesor, E es el módulo de elasticidad, ν es el coeficiente de Poisson y ρ es la densidad.

En la tabla 4.5 se muestran los parámetros de convergencia usados en la optimización para cada uno de los algoritmos de optimización.

Tabla 4.5: Tabla de parámetros de convergencia algoritmos de optimización.

Parámetros	Valor
ϵ_1	10e-6
ϵ_2	10e-3
n_g	10

El parámetro ϵ_1 corresponde al criterio de convergencia para cada población dentro del marco de algoritmos GA y MBPSO, ϵ_2 es el criterio de convergencia para las 4 poblaciones usadas en MpGA y n_g es el número de generaciones de diferencia para comprobar la convergencia de los algoritmos de optimización.

En la tabla 4.6 y 4.7 se presentan los hiperparámetros usados en la optimización de bandgaps usando GA y PSO respectivamente.

Tabla 4.6: Tabla resumen de hiperparámetros utilizados en GA

Hiperparámetro	Valor
n_{genes}	32 - 136
p_{size}	32
p_c	0.95
p_m	0.05

Tabla 4.7: Tabla resumen de hiperparámetros utilizados en MBPSO

Hiperparámetro	Valor
n_{genes}	32 - 136
t_p	32
c_1	2
c_2	2
v_{max}	10

Donde n_{genes} corresponde al número de genes o características con las que se definen las placas, los valores 32 y 136 corresponden al número de genes para los casos de placas con 16 y 32 elementos por arista respectivamente, p_{size} y t_p son el número de individuos y partículas usadas en la optimización respectivamente, p_c es la probabilidad de crossover, p_m es la probabilidad de mutación, c_1 y c_2 son los términos de inercia según la ecuación 2.42 y v_{max} es la velocidad máxima según la ecuación B.2.

En el modelado de placas en IGA se usan dos parámetros adicionales p y q , los cuales indican el orden las funciones usadas para la discretización de la solución, para este trabajo se usa $p = q = 3$.

Para todo este trabajo y para todos los casos se realiza la optimización topológica calculando 6 frecuencias normales y al maximizar el bandgap entre las frecuencias 3 y 4.

4.2.2. Maximización de Bandgaps en placas cuadradas

En esta sección se muestran los resultados referentes a la optimización de bandgaps para placas cuadradas, usando los parámetros mostrados en las tablas 4.4 y 4.5. En las tablas 4.8 a la 4.11 se presentan la comparación entre tiempos de convergencia y valores de bandgaps para las diferentes combinaciones entre algoritmos de optimización MBPSO y MpGA y para modelos FEA e IGA, variando la cantidad de elementos por arista usados, también se muestran los valores promedios y desviaciones estándar calculados a partir de 10 repeticiones de cada optimización.

Tabla 4.8: Tabla comparativa entre algoritmos MBPSO y MpGA de Bandgaps máximos y tiempos de optimización para placa de 16 elementos por arista modelada con FEA.

	MBPSO		**MpGA**	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	1108.21	63.73	8411.36	50.94
2	1883.81	-17.72	5442.66	2.90
3	3055.41	75.12	8399.21	23.16
4	1746.41	79.91	7134.78	16.35
5	1422.85	-3.11	5682.87	0.00
6	1893.70	63.73	9241.98	50.94
7	986.63	0.00	5089.85	0.00
8	2352.17	45.15	3931.65	0.00
9	1944.81	6.41	4271.92	0.00
10	1661.25	-36.80	9071.21	59.15
Promedio	1805.52	27.64	6667.74	20.34
Desviación Estándar	598.32	42.51	2023.37	24.40

Tabla 4.9: Tabla comparativa entre algoritmos MBPSO y MpGA de Bandgaps máximos y tiempos de optimización para placa de 32 elementos por arista modelada con FEA.

	MBPSO		**MpGA**	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	41961.40	124.73	191499.20	6.23
2	82172.62	86.68	268474.75	0.00
3	53132.80	28.26	201350.80	38.16
4	91058.21	22.63	263009.51	0.00
5	64903.68	31.64	230652.67	0.86
6	89688.48	145.64	204598.58	20.26
7	47888.98	7.62	230472.48	-6.33
8	74917.80	104.96	230055.78	-16.93
9	73710.81	6.84	225734.97	21.03
10	46658.92	68.07	247897.51	-57.40
Promedio	66609.37	62.70	229374.62	0.58
Desviación Estándar	18356.31	50.58	25502.73	25.74

Tabla 4.10: Tabla comparativa entre algoritmos MBPSO y MpGA de Bandgaps máximos y tiempos de optimización para placa de 16 elementos por arista modelada con IGA.

	MBPSO		**MpGA**	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	307.19	0.00	2089.97	41.32
2	303.63	41.32	1422.49	0.00
3	932.69	-19.91	1244.45	0.00
4	756.87	-7.02	2462.38	41.32
5	435.38	41.32	2057.74	0.00
6	616.98	0.00	1288.26	0.00
7	580.35	24.44	2896.32	47.32
8	343.81	-9.06	1872.18	24.44
9	425.02	-8.92	2666.44	56.14
10	330.66	-123.44	2997.65	28.84
Promedio	503.25	-6.12	2099.78	23.93
Desviación Estándar	213.91	46.52	649.42	22.37

Tabla 4.11: Tabla comparativa entre algoritmos MBPSO y MpGA de Bandgaps máximos y tiempos de optimización para placa de 32 elementos por arista modelada con IGA.

	MBPSO		**MpGA**	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	11477.96	16.24	50516.59	0.00
2	11521.86	24.79	55697.52	74.10
3	13330.36	72.29	53709.42	-40.48
4	8330.87	-45.41	53171.06	-24.51
5	26785.22	106.64	54592.02	141.03
6	12919.89	73.84	52583.43	0.00
7	11473.61	33.45	53726.16	50.68
8	9855.12	39.37	53559.25	0.00
9	16438.19	93.50	51987.40	-38.07
10	11231.10	79.66	54457.22	-24.60
Promedio	13336,41	49,43	53400,00	13,81
Desviación Estándar	5190.68	45.10	1455.77	58.05

Luego de analizar las tablas 4.8 a la 4.11 las cuales resumen los resultados de la optimización topológica para cada caso de estudio, comparando de acuerdo a los algoritmos de optimización utilizados, se obtiene que, en promedio, el algoritmo de optimización MpGA es siempre mas lento en comparación a MBPSO, demorándose entre 3,44 y 4,17 veces más, este comportamiento es esperable debido a la naturaleza de MpGA, donde se trabaja con poblaciones en paralelo, las cuales evolucionan de forma separada, y cada una contiene el mismo número de individuos que en MBPSO, luego el número total de individuos con los que se trabaja en MpGA es 4 veces mayor al de MBPSO.

Al comparar los valores de los bandgaps se tiene que, para todos los casos de comparación, a excepción de IGA con 16 elementos por arista, el algoritmo MBPSO obtiene mejores resultados promedios. Este comportamiento sugiere que los resultados luego de optimizar con MBPSO tienen a ser mayores a los entregados por MpGA, pero para poder realizar la afirmación se requiere que se realicen un mayor número de repeticiones de las optimizaciones, esto es debido a la excepción antes mencionada. También se obtiene que, dado que existen valores no positivos entre los resultados de las optimizaciones, ninguno de los algoritmos son ajenos a caer y quedarse atrapados en máximos locales.

Cuando se comparan los resultados variando solo el número de elementos que conforman la placa, se obtiene que estos mejoran en el caso de MBPSO y empeoran en el caso de MpGA, esto sugiere que el primer algoritmo es capaz de trabajar de manera más efectiva con un mayor número de variables (o en un espacio de dimension mayor) en comparación al segundo. Además, los tiempos de optimización aumentan en diferentes proporciones al aumentar el número de elementos, para el caso de FEA aumentan entre 34.40 y 36,89 veces, mientras que para IGA aumentan entre 25.43 y 26.50 veces.

Comparando los resultados variando solo el algoritmo con el que se modelan las placas, se obtiene que no hay un patrón a seguir al analizar los valores para los bandgaps, en cambio al comparar los tiempos se obtiene que optimizar con IGA es siempre mas rápido que con FEA, habiendo también diferencias de acuerdo al número de elementos usados en el modelado, donde se tiene que optimizar con FEA es entre 3.17 y 3.58 veces mas lento para el caso de placas con 16 elementos por arista y entre 4.29 y 4.99 veces para el caso de placas con 32 elementos por arista.

En la figura 4.5, figuras 4.6 a la 4.13, figuras 4.14 a la 4.19 se presentan las topologías, diagramas de bandas y curvas de convergencia respectivamente para los mejores resultados de cada una de las 10 repeticiones para cada uno de las combinaciones mostradas en las tablas presentadas anteriormente.

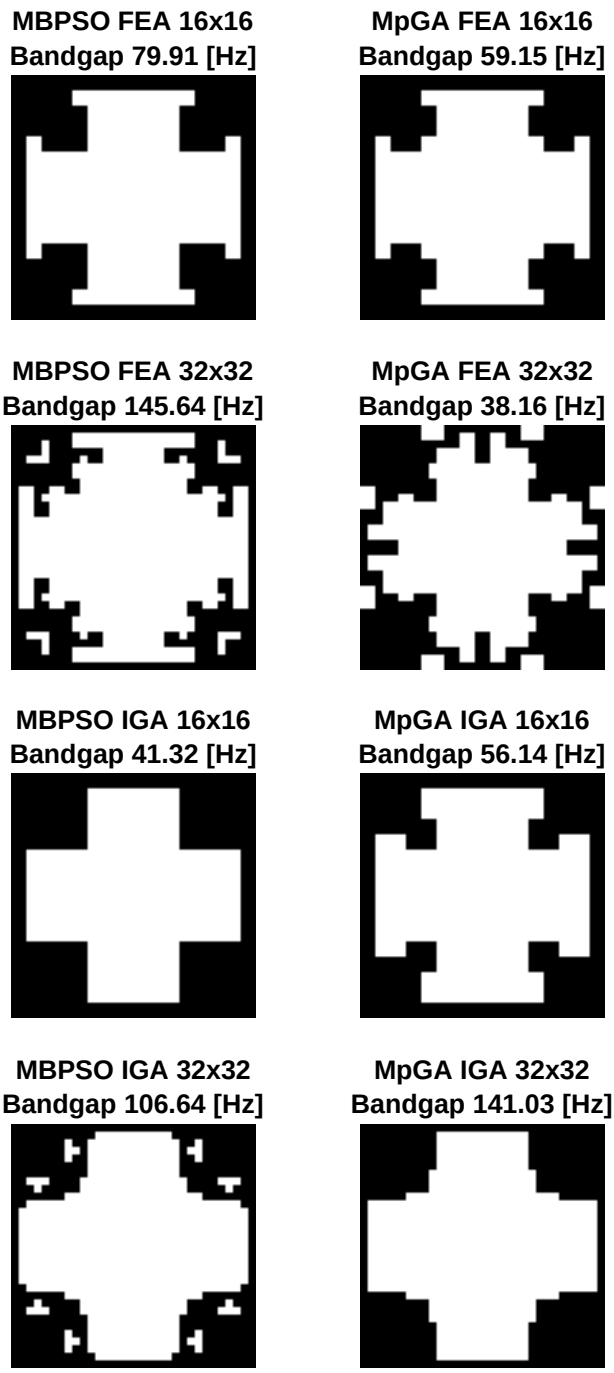


Figura 4.5: Topologías con mayores valores de bandgaps para diferentes combinaciones de algoritmos de optimización, algoritmos para el modelado de placas y número de elementos máximos en cada placa. El color blanco representa el vacío, mientras que el color negro representa el material construido.

La figura 4.5 muestra las topologías que representan las placas con mayores bandgaps para cada uno de los casos analizados, donde, se aprecia que las topologías resultantes son muy similares entre si, dado que cada una cuenta con aproximadamente la misma región vacía (región blanca), y por ende una región de material construido similar (región negra).

Es importante señalar que pequeñas diferencias en las topologías de las placas pueden significar diferencias mas significativas (del mismo orden de magnitud que el bandgap), el mejor ejemplo de lo anterior es la comparación de las primeras dos topologías, donde estas son idénticas salvo que la segunda quita un elemento de placa por esquina de la cruz formada por los elementos vacíos, en esta comparación la diferencia en el valor del bandgap es de 20.76 [Hz]. También, se tiene que los valores de bandgaps que se pueden obtener con placas de 16x16 elementos son considerablemente menores a los de placas con 32x32 elementos, esto es debido a que las últimas poseen más elementos y por tanto mas combinaciones potencialmente mejores.

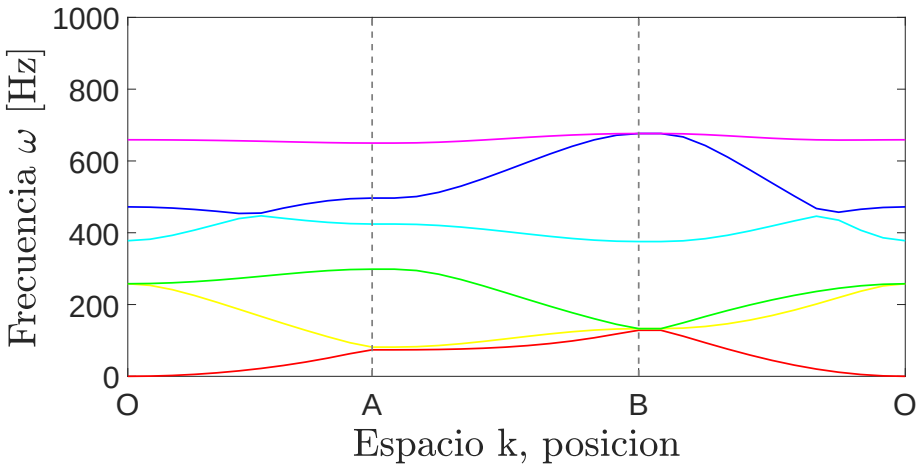


Figura 4.6: Diagrama de bandas resultante de optimización utilizando algoritmos MBPSO y FEA para una placa de 16 elementos por arista.

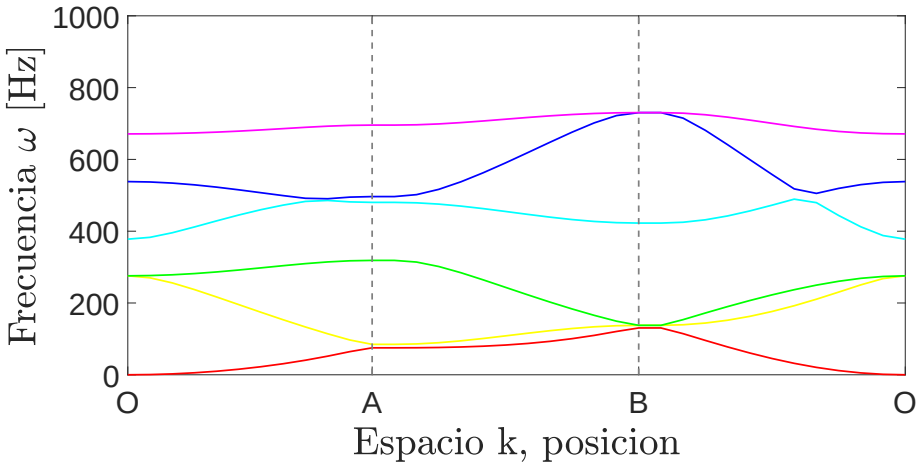


Figura 4.7: Diagrama de bandas resultante de optimización utilizando algoritmos MpGA y FEA para una placa de 16 elementos por arista.

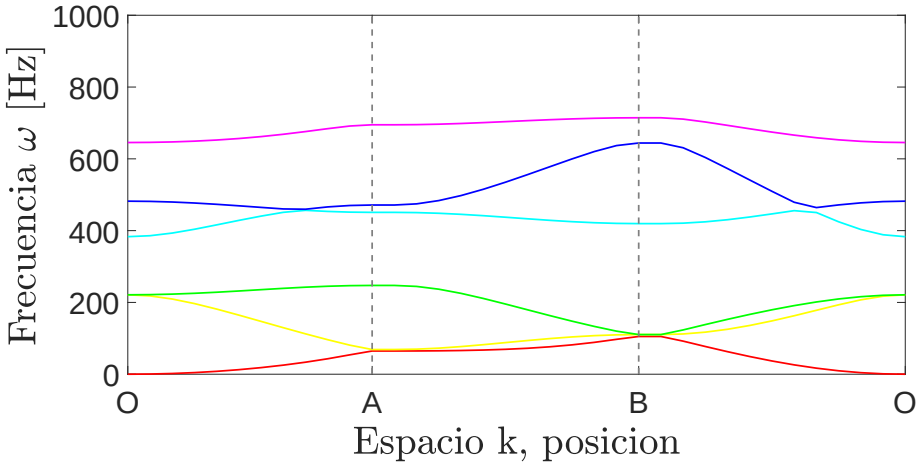


Figura 4.8: Diagrama de bandas resultante de optimización utilizando algoritmos MBPSO y FEA para una placa de 32 elementos por arista.

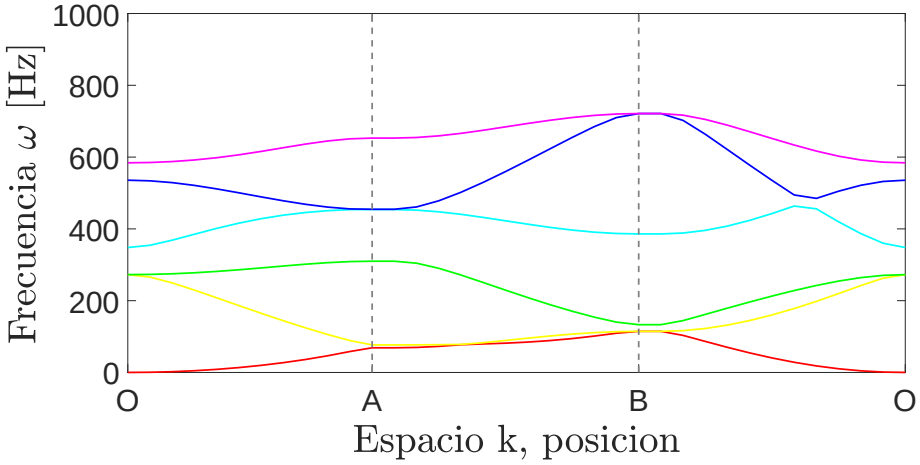


Figura 4.9: Diagrama de bandas resultante de optimización utilizando algoritmos MpGA y FEA para una placa de 32 elementos por arista.

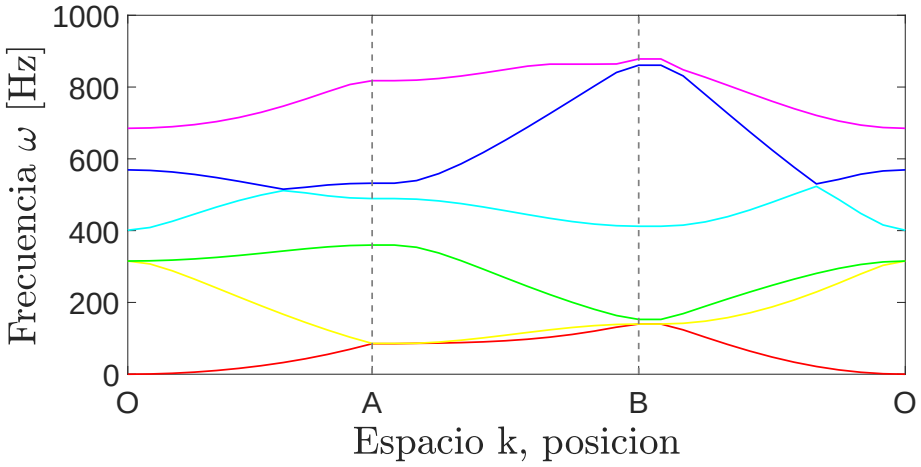


Figura 4.10: Diagrama de bandas resultante de optimización utilizando algoritmos MBPSO e IGA para una placa de 16 elementos por arista.

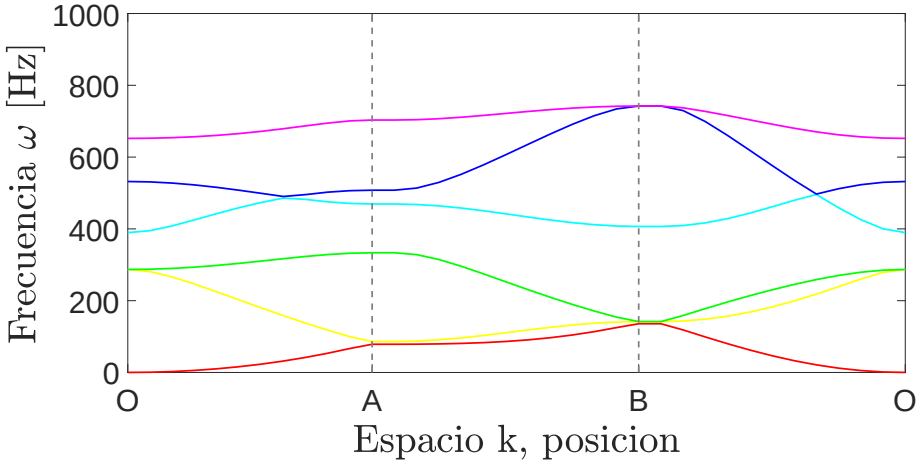


Figura 4.11: Diagrama de bandas resultante de optimización utilizando algoritmos MpGA e IGA para una placa de 16 elementos por arista.

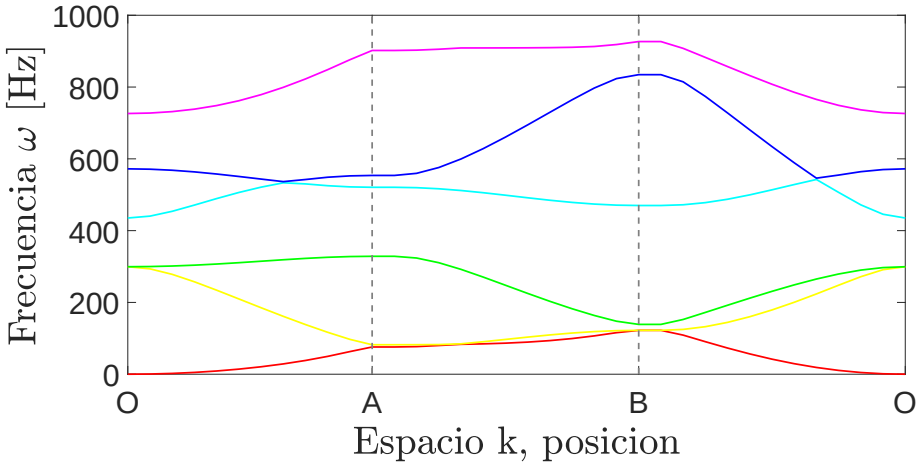


Figura 4.12: Diagrama de bandas resultante de optimización utilizando algoritmos MBPSO e IGA para una placa de 32 elementos por arista.

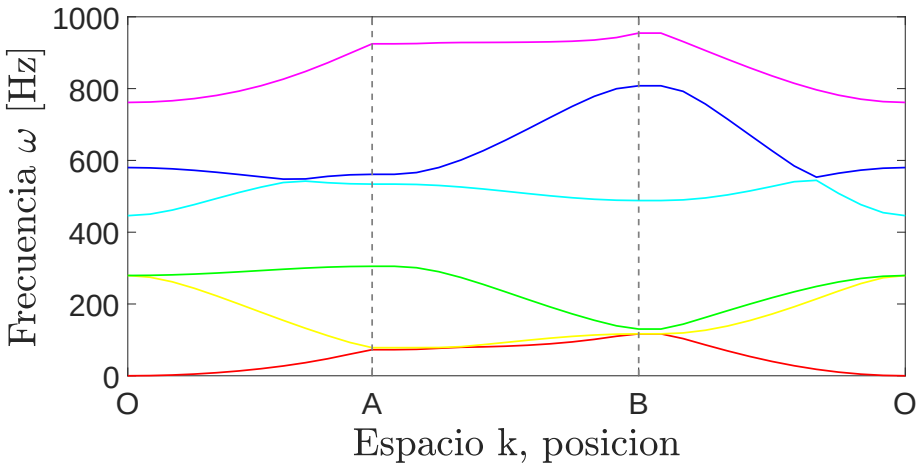


Figura 4.13: Diagrama de bandas resultante de optimización utilizando algoritmos MpGA e IGA para una placa de 32 elementos por arista.

Respecto a las figuras 4.6 a la 4.13 referentes a los diagramas de bandas, se observa que los diagramas son semejantes, teniendo los bandgaps en los mismos rangos de frecuencia. También las frecuencias asociadas a los modos normales tienen comportamientos parecidos, por ejemplo, las frecuencia asociadas al primer y quinto modo normal siempre tiene un máximo en el punto **B** del recorrido para el vector de onda. La mayor diferencia en los diagramas es la asociada al sexto modo normal, donde en ciertos casos esta no se apega el quinto modo en el punto **B** del diagrama. Se observa además que, a mayor valor del bandgap, las frecuencias asociadas a los 3 últimos modos normales tienden a aumentar sus valores en el diagrama, mientras que las frecuencias asociadas a las primeras 3 frecuencias normales tienden a tener los mismos.

Haciendo una comparación con la figura 4.3, la cual muestra el diagrama de bandas de una placa completa, se aprecia que hay cambios mínimos para los primeros 3 modos, ya

que se presentan los mismos comportamientos y valores similares a lo largo del diagrama, en cambio, existen mayores diferencias en los últimos 3 modos, donde en todos los casos los modos 3 y 4 dejan de se idénticos en la mayor parte del diagrama para formar la banda, los modos 5 y 6 también se separan entre si y cada uno tiene valores de frecuencia menores, las cuales siempre se siguen maximizando en el punto **B**.

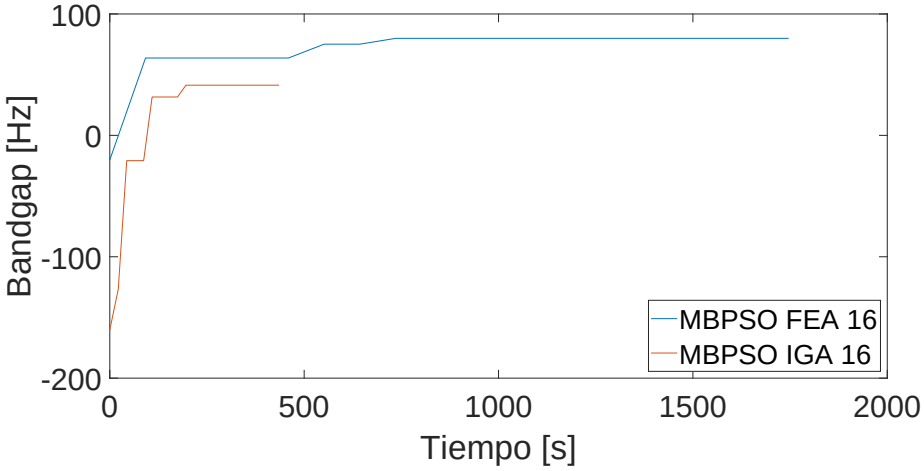


Figura 4.14: Comparación de curvas de convergencia resultante entre modelos IGA y FEA, utilizando algoritmo MBPSO para una placa de 16 elementos por arista.

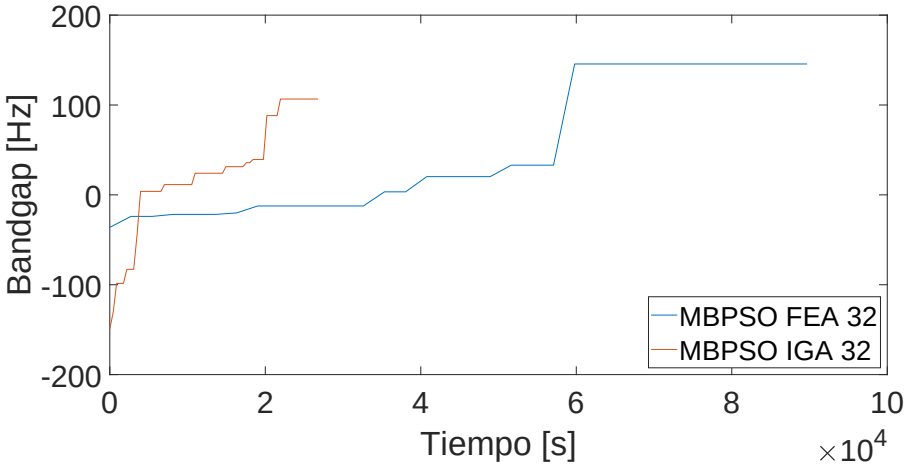


Figura 4.15: Comparación de curvas de convergencia resultante entre modelos IGA y FEA, utilizando algoritmo MBPSO para una placa de 32 elementos por arista.

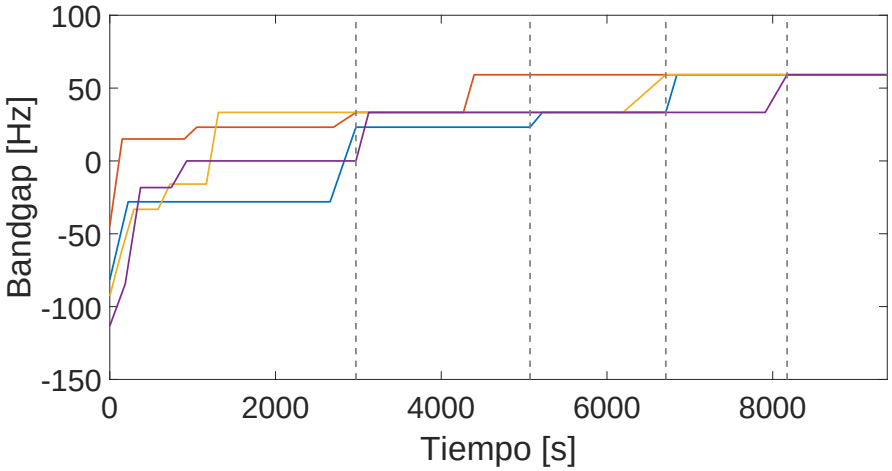


Figura 4.16: Curva de convergencia resultante de optimización utilizando algoritmos MpGA y FEA para una placa de 16 elementos por arista.

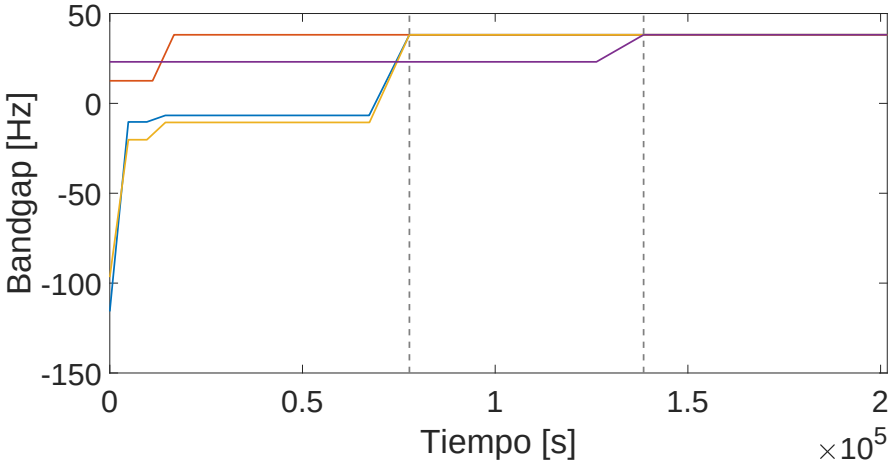


Figura 4.17: Curva de convergencia resultante de optimización utilizando algoritmos MpGA y FEA para una placa de 32 elementos por arista.

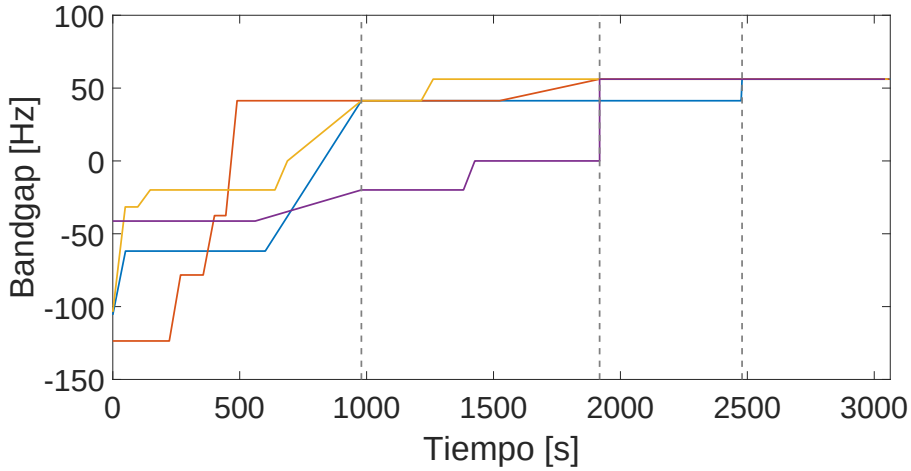


Figura 4.18: Curva de convergencia resultante de optimización utilizando algoritmos MpGA e IGA para una placa de 16 elementos por arista.

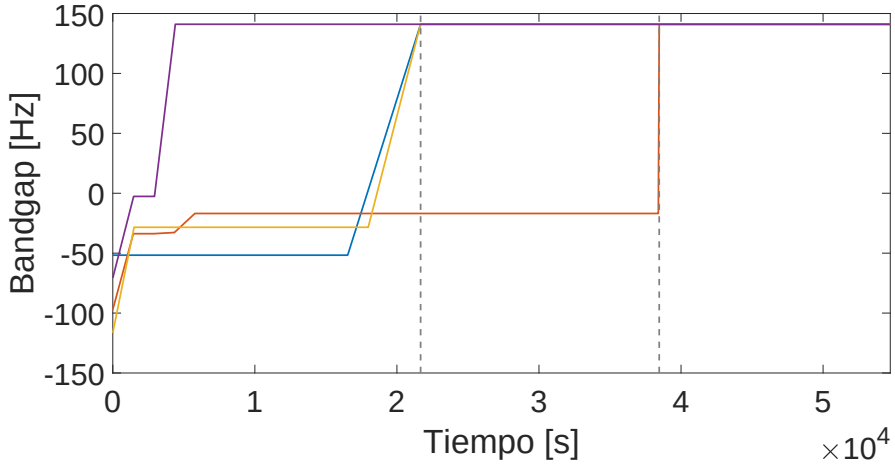


Figura 4.19: Curva de convergencia resultante de optimización utilizando algoritmos MpGA e IGA para una placa de 32 elementos por arista.

En las figuras 4.14 a la 4.19 correspondientes a las curvas de convergencia, se observa que para todos los casos mostrados, los resultados no dependen de las condiciones iniciales para los individuos o partículas, o sea que es independiente de la población inicial con la que parte cada algoritmo, esto se debe a que a pesar de que tan alto (o bajo) sea el mayor fitness de la población inicial, esta tiene la suficiente variabilidad para que se realice una optimización satisfactoria.

Al comparar las curvas de convergencia resultantes es fácil notar que los tiempos de optimización totales para el caso de MpGA son mayores a los de MBPSO, los cuales son comparables (pero siempre menores) a los tiempos hasta la primera convergencia de cada población para algoritmos genéticos, denotado por la primera linea vertical punteada. De las figuras 4.14 y 4.15 es fácil notar la diferencia en tiempos de optimización al comparar entre

modelos, donde IGA siempre es más rápido que FEA, independiente del número elementos con los que se modele la placa.

Comparando los valores de bandgaps, para el caso de MpGA se tiene que al terminar la primera convergencia de cada población los resultados tienden a ser menores a los arrojados por MBPSO, a pesar de demorarse tiempos similares. Lo anterior reafirma la necesidad de trabajar con algoritmos genéticos de varias poblaciones, ya que realizar la optimización con una población no es suficiente y no garantiza resultados satisfactorios.

Dada la gran variabilidad en los resultados es que se realizan 300 optimizaciones para la combinación MBPSO-IGA y placas de 16 elementos por arista, ya que esta es la combinación que se demora menos tiempo en promedio. La tabla 4.12 muestra el promedio y desviación estándar de los tiempos de optimización y bandgap resultantes, de la cual se obtiene que los valores para el tiempo son similares a cuando se realizaron 10 optimizaciones, en cambio para el caso del bandgap se tiene que el promedio es mayor y la desviación estándar es menor al caso anterior.

Tabla 4.12: Tabla resumen de promedio y desviación estándar de 300 iteraciones para combinación MBPSO-IGA en una placa de 16x16 elementos.

	Tiempo [s]	Bandgap [Hz]
Promedio	507.39	25.92
Desviación Estándar	185.93	29.67

De los datos obtenidos de las sucesivas optimizaciones se crear diferentes gráficos, la figura 4.20 muestra el valor del bandgap en función del número de iteraciones según el orden en que fueron obtenidas, se grafíca también el valor del promedio. De este gráfico se observa el comportamiento errático del bandgap, el cual parece seguir ninguna regla, solo se obtiene que los casos donde los bandgap son menores -50 [Hz] pueden considerarse como outliers.

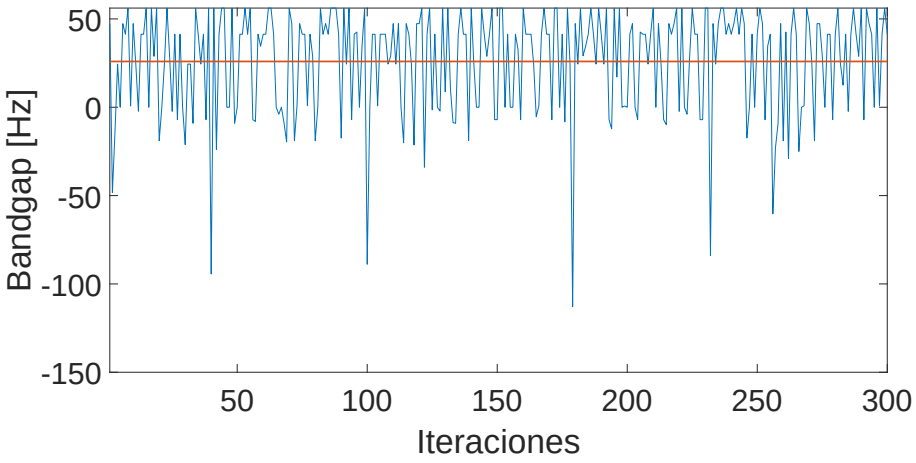


Figura 4.20: Bandgap en función del número de iteraciones.

La figura 4.21 muestra el bandgap ordenado de menor a mayor en función del número de iteraciones, se grafíca también el valor del promedio. De esta figura se puede ver claramente

que se supera el promedio mas de un 50.00 % de las veces mientras que se tienen valores negativos menos de un 33.33 % de las veces.

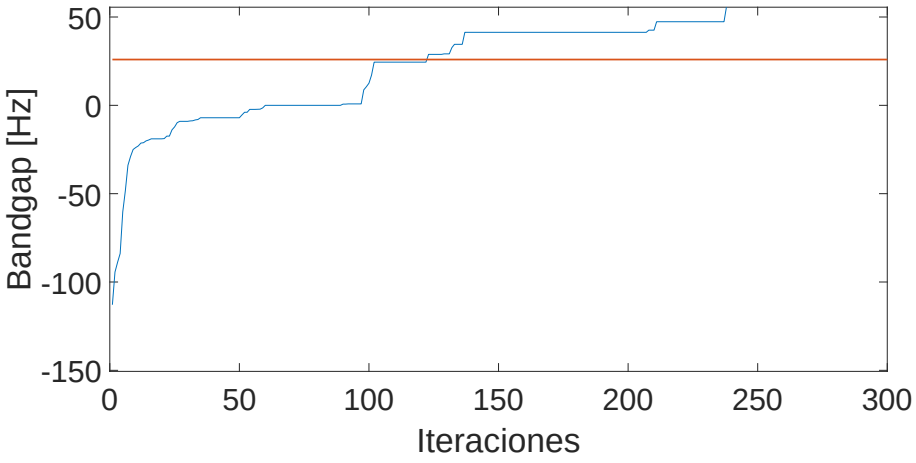


Figura 4.21: Bandgap ordenado de menor a mayor en función del número de iteraciones.

Se crea un histograma de los datos y se ajustan las dos distribuciones que se adecuan de mejor manera a los datos, las figuras 4.22 a la 4.24 muestran la densidad de probabilidad, probabilidad acumulada y gráfico de probabilidad respectivamente.

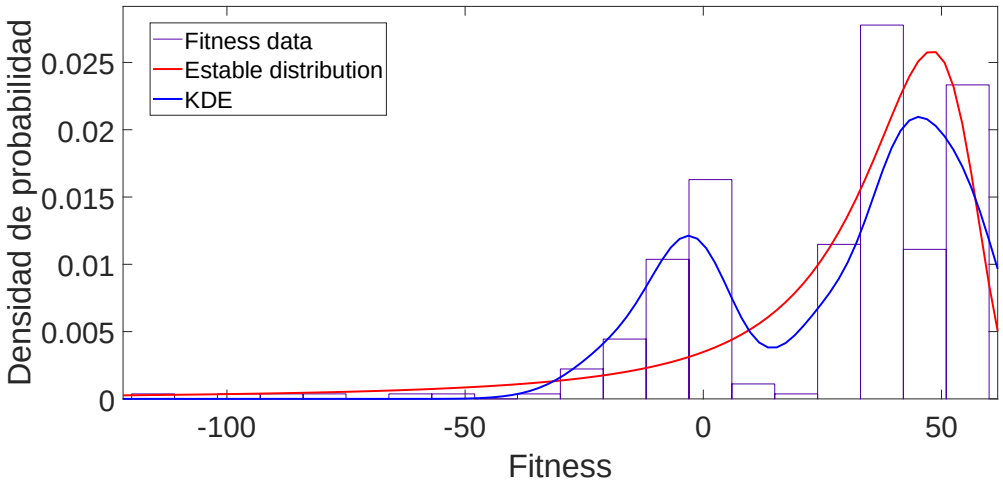


Figura 4.22: Grafico de densidad de probabilidad luego de ajuste según el histograma.

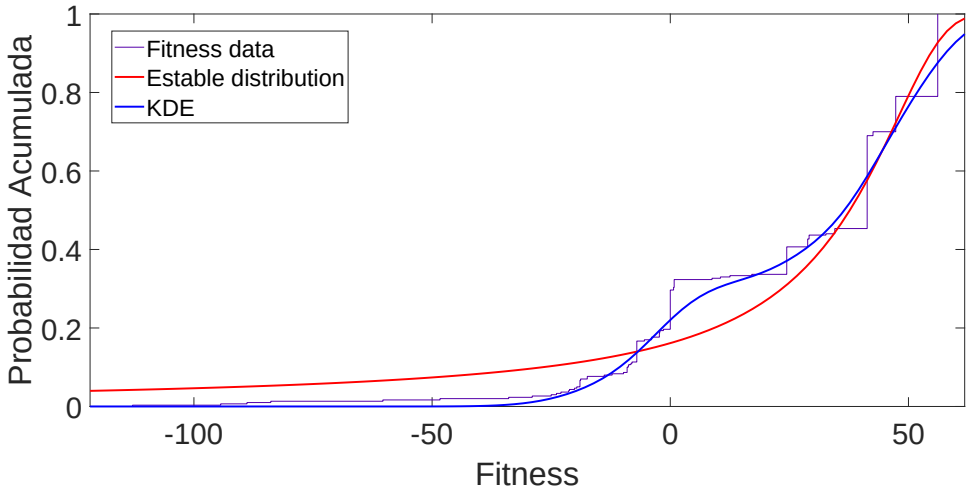


Figura 4.23: Grafico de probabilidad acumulada luego de ajuste según el histograma.

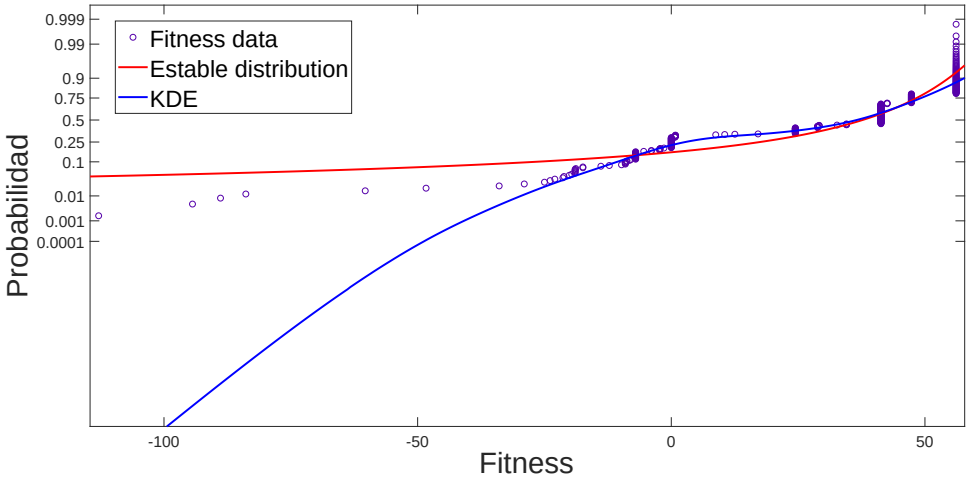


Figura 4.24: Grafico de probabilidad luego de ajuste según el histograma.

Las dos distribuciones que mas se ajustan a los datos son la Distribución Estable y la Estimación de densidad por Kernel (KDE por sus siglas en inglés), donde se usa un kernel gaussiano. La distribución estable ajusta mejor los datos mayores, mientras que KDE se ajusta mejor de forma promedio. Esto último aprecia al analizar el gráfico de probabilidad acumulada y en el gráfico de probabilidad, donde se ve que la distribución KDE sigue de mejor manera los datos de manera promedio y la distribución estable se ajusta mejor para en el último tramo de la curva.

Al analizar la función de probabilidad acumulada se tiene que la probabilidad de obtener ciertos valores cambia de acuerdo al ajuste a considerar, la tabla 4.13 muestra algunos valores, donde se tiene que, para ambos casos (en especial para la distribución estable), la probabilidad de llegar al valor óptimo de la distribución es baja, pero no así de obtener un valor alto, por

ejemplo 40 [Hz].

Tabla 4.13: Valores de probabilidad acumulada para diferentes valores de fitness.

	Estable	KDE
F(x=0)	0.1614	0.2203
F(x=10)	0.2040	0.3020
F(x=20)	0.2694	0.3445
F(x=30)	0.3749	0.4170
F(x=40)	0.5472	0.5606
F(x=50)	0.7920	0.7649
F(x=56.14)	0.9275	0.8758
F(x=60)	0.9765	0.9294

En las tablas 4.14 y 4.15 se muestran los parámetros que definen las distribuciones.

Tabla 4.14: Parámetros distribución Estable

	Alpha	Beta	Gamma	Delta
Distribución Estable	1.0454	-1.0000	10.9031	43.6163

Tabla 4.15: Parámetros distribución KDE

	Ancho de Banda
KDE	7.4646

4.2.3. Maximización de Bandgaps en placas cuadrada con orificio circular

Antes de realizar la optimización topológica en placas con orificio circulares se parte por determinar como varían los valores de bandgaps para diferentes radios de orificios sin hacer cambios en la topología, ver figura 4.25.

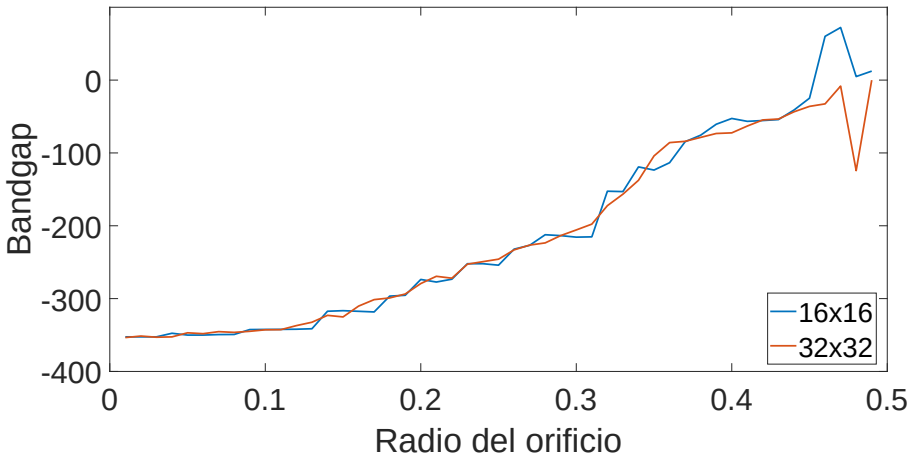


Figura 4.25: Variación de valor de bandgaps para diferentes radios.

La figura 4.25 muestra que el valor del bandgap presenta una tendencia creciente conforme el radio del orificio aumenta. El determinar este comportamiento previo a la realización de la optimización topológica es importante porque indica que se puede aumentar el valor para bandgap solo agregando un orificio, sin cambiar la topología misma de la placa.

Es importante señalar que se realizan optimizaciones para un número discreto de radios para ambos casos de placas utilizadas, esto es debido a que, en caso de utilizarse en rango de radios muy grande, se utilizaría un tiempo mayor al disponible para realizare este trabajo. Los radios fueron escogidos de tal forma que cada uno sea múltiplo entero de la diagonal de un elemento que conforma cada placa, luego para cada placa se tiene un número predefinido de radios con los cuales trabajar.

Las dos curvas tienen un comportamiento similar salvo para valores de radio superiores a 0.43 [m] y 0.46 [m] para placas de 16x16 y 32x32 elementos respectivamente. A pesar de que se tengan radios diferentes en cada caso, estos tienen en común que ambos forman una placa con un espesor (distancia entre el borde de la placa y el borde del orificio) menor al tamaño de un elemento que conforma la misma. Este comportamiento se puede interpretar como que IGA pierde precisión en el cálculo de frecuencias cuando el espesor de la placa es menor al tamaño de un elemento que la compone.

Luego de determinar el comportamiento del bandgap con respecto al radio del orificio se procede a realizar la optimización topológica, para esto se usan placas de 16 y 32 elementos por arista y se realiza el procedimiento para una serie de radios predeterminados.

En las tablas 4.16 a 4.18 y 4.19 a 4.24 se presentan los resultados referentes a la optimización topológica, maximizando bandgaps en placas cuadradas con orificios circualres para diferentes radios de orificios y para placas de 16x16 elementos y 32x32 elementos respectivamente. Se realizan 10 repeticiones de cada optimización al igual que en los otros casos de estudio.

Tabla 4.16: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 16x16 elementos para radios de 0.0894 [m] y 0.1778 [m].

	$r_1 = 0.0894$ [m]		$r_2 = 0.1778$ [m]	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	951.76	-94.02	645.51	48.44
2	639.42	-82.22	973.04	59.27
3	524.55	-57.44	694.26	60.77
4	455.40	-68.20	514.70	60.77
5	884.54	-58.73	719.02	60.77
6	875.70	-57.44	928.52	59.21
7	981.96	-77.65	596.48	48.44
8	723.24	-106.78	566.36	-46.45
9	825.03	-61.74	931.27	60.77
10	462.53	-71.15	740.32	48.44
Promedio	732.41	-73.54	730.95	46.04
Desviación Estandar	201.15	16.76	172.66	32.98

Tabla 4.17: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 16x16 elementos para radios de 0.2662 [m] y 0.3546 [m]

	$r_3 = 0.2662$ [m]		$r_4 = 0.3546$ [m]	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	1099.23	9.65	495.73	81.21
2	774.18	4.06	565.08	81.21
3	650.30	9.65	663.86	86.53
4	480.46	4.06	564.54	81.21
5	514.02	9.65	564.22	81.21
6	515.11	9.65	459.85	81.21
7	965.67	44.75	459.01	81.21
8	1015.60	-44.25	460.52	81.21
9	585.08	4.06	538.53	81.21
10	617.71	9.65	462.96	81.21
Promedio	721.74	6.09	523.43	81.75
Desviación Estandar	228.47	21.35	68.00	1.68

Tabla 4.18: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 16x16 elementos para radios de 0.4429 [m].

	$r_5 = 0.4429$ [m]	
Repetición	**Tiempo**	**Bandgap**
1	480.31	102.13
2	477.77	102.13
3	476.86	102.13
4	483.31	102.13
5	479.82	102.13
6	480.61	102.13
7	479.77	102.13
8	483.93	102.13
9	484.21	102.13
10	484.37	102.13
Promedio	481.10	102.13
Desviación Estandar	2.72	0.00

Las tablas 4.16 a la 4.18 muestran los resultados de la optimización topológica para diferentes valores de radios en placas de 16x16 elementos. Comparando los resultados promedio se tiene que estos no siempre son crecientes, a pesar del comportamiento encontrado en la figura 4.25, específicamente se encuentra que para el caso del radio r_3 el valor del bandgap promedio es menor al del radio anterior. Para los radios r_4 y r_5 se tiene que en todos los casos (salvo uno) siempre se llega al mismo resultado para cada radio.

Al comparar los tiempos de optimización, se encuentra que no existe una tendencia para radios menores o iguales a r_3, por otra parte, para radios mayores a r_3, los tiempos promedio como sus desviaciones estándar disminuyen considerablemente, esto coincide con el comportamiento encontrado para los valores de bandgaps para los mismos radios. También ,se encuentra que no existe relación entre tiempo de optimización y valores de bandgap, o sea, un mayor tiempo de optimización no asegura encontrar un mejor resultado.

Tabla 4.19: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 32x32 elementos para radios de 0.0452 [m] y 0.0894 [m]

	$r_1 = 0.0452$ [m]		$r_2 = 0.0894$ [m]	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	10539.63	-30.12	13763.45	-19.59
2	16834.34	50.41	10287.81	2.16
3	6901.84	-63.97	14016.70	-22.28
4	12958.14	-34.74	7254.53	-38.52
5	5894.72	-15.04	9755.76	2.90
6	13105.02	-10.46	7748.74	-40.41
7	5604.54	-41.86	8535.23	10.96
8	6895.36	-34.01	9820.27	-3.25
9	11186.84	-49.87	19644.53	5.37
10	8295.48	-36.44	16330.39	-21.73
Promedio	9821.59	-26.61	11715.74	-12.44
Desviación Estandar	3723.76	31.14	4067.17	18.54

Tabla 4.20: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 32x32 elementos para radios de 0.1336 [m] y 0.1778 [m]

	$r_3 = 0.1336$ [m]		$r_4 = 0.1778$ [m]	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	6486.72	-58.53	12772.16	-14.26
2	12605.51	-33.42	11100.32	-20.99
3	9503.51	-23.93	8568.77	-33.39
4	15525.43	-3.81	10075.73	-7.45
5	8589.19	0.82	11972.02	10.07
6	10564.89	-57.37	7462.11	-0.08
7	8586.29	-12.08	8911.53	-22.73
8	6821.14	-30.35	13890.14	3.80
9	7990.42	-26.07	14174.42	-13.68
10	8654.23	-39.12	17528.52	-28.86
Promedio	9532.73	-28.39	11645.57	-12.76
Desviación Estandar	2748.49	20.11	3066.35	14.30

Tabla 4.21: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 32x32 elementos para radios de 0.2220 [m] y 0.2662 [m]

	$r_5 = 0.2220$ [m]		$r_6 = 0.2662$ [m]	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	17901.38	6.13	7416.69	93.96
2	14241.43	62.46	8373.79	19.49
3	7299.99	37.28	9512.49	9.23
4	8982.88	51.13	8104.60	1.53
5	12633.28	5.59	11448.15	48.47
6	12256.48	11.80	19975.32	135.59
7	21578.86	-6.52	9538.09	71.35
8	11321.72	15.01	14158.79	79.26
9	13226.10	6.58	14319.30	152.43
10	8118.70	30.39	12681.27	141.94
Promedio	12756.08	21.98	11552.85	75.32
Desviación Estandar	4404.24	22.40	3858.52	55.82

Tabla 4.22: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 32x32 elementos para radios de 0.3104 [m] y 0.3546 [m]

	$r_7 = 0.3104$ [m]		$r_8 = 0.3546$ [m]	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [s]**	**Bandgap [Hz]**
1	7605.85	141.73	18802.08	177.92
2	9206.76	19.61	7237.49	48.79
3	9429.99	142.68	7294.25	174.63
4	24229.42	203.13	7492.39	196.58
5	10345.41	156.00	6957.38	179.78
6	18499.27	175.44	12482.22	172.19
7	9660.26	152.88	10558.54	198.68
8	11729.38	148.66	14510.33	54.38
9	13343.39	161.18	14183.39	193.67
10	17216.90	212.42	14930.84	197.70
Promedio	13126.66	151.37	11444.89	159.43
Desviación Estandar	5271.80	52.33	4159.70	57.72

Tabla 4.23: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 32x32 elementos para radios de 0.3987 [m] y 0.4429 [m]

	$r_9 = 0.3987$ [m]		$r_{10} = 0.4429$ [m]	
Repetición	**Tiempo [s]**	**Bandgap [Hz]**	**Tiempo [m]**	**Bandgap [Hz]**
1	10305.78	222.57	6349.30	251.97
2	9845.73	215.21	8142.54	215.76
3	12022.43	226.68	6917.22	215.10
4	10407.62	220.01	9867.53	239.63
5	12084.01	218.13	10274.22	238.79
6	11335.18	217.18	8438.81	247.88
7	6071.65	226.68	9643.74	218.54
8	11446.10	235.01	8455.15	251.97
9	13463.27	182.64	10226.83	249.57
10	9954.81	187.30	13145.72	238.65
Promedio	10693.66	215.14	9146.11	236.78
Desviación Estandar	1977.86	16.94	1943.09	14.94

Tabla 4.24: Tabla resumen de valores de bandgaps y tiempos de optimización para placa de 32x32 elementos para radios de 0.4871 [m].

	$r_{11} = 0.4871$ [m]	
Repetición	**Tiempo** [s]	**Bandgap** [Hz]
1	7176.65	0.00
2	7649.48	-0.01
3	8215.48	-0.01
4	7262.20	0.00
5	7383.70	0.00
6	6424.10	0.00
7	5976.14	0.00
8	8168.52	0.00
9	6337.61	0.00
10	7130.37	0.00
Promedio	7172.43	0.00
Desviación Estandar	749.09	0.00

Las tablas 4.19 a la 4.24 muestran los resultados de la optimización topológica para diferentes valores de radios en placas de 32x32 elementos. Comparando los resultados de los bandgap promedios se tiene que para estos casos tampoco se tiene una tendencia creciente, esto es especialmente claro para radios menores o iguales a r_4 y para r_{11}, para el resto de radios analizados si se tiene un comportamiento creciente. Los valores encontrados para r_{11} son cercanos a cero en todos los casos, esto se puede explicar ya que este radio de orificio es

mayor al radio para el cual IGA pierde precisión en este tipo de placas.

Al comparar los tiempos promedio y desviaciones estándar, se tiene que tampoco existe una tendencia clara, para el caso de r_{11} se encuentra que el tiempo promedio es considerablemente menor al resto, esto sugiere que el algoritmo converge más rápidamente para este valor de radio de orificio. Al igual que para placas de 16x16 elementos, mayores tiempos de optimización no aseguran mejores resultados.

Comparando solo los mejores resultados de la optimización topológica entre placas cuadradas y placas cuadras con orificios circulares, ambas utilizando IGA y MBPSO, se tiene que para una placa con orificio de 16 elementos por arista el tiempo de optimización es 1.57 veces mayor y el bangdap es 2.47 veces mayor en comparación a una placa regular. Para el caso de una placa con orificio de 32 elementos por arista se tiene que el tiempo es 0.23 veces menor y el bandgap es 2.26 veces mayor.

En ambos casos se aumenta en valor del bandgap resultante, para placas de 16 elementos por arista se aumenta el tiempo de optimización, pero debido a que inicialmente los tiempo son bajos, este aumento no se considera problemático. Para placas de 32 elementos por arista el tiempo se reduce enormemente, ya que se reduce en más de un orden de magnitud.

Al analizar los valores promedios se encuentra que los bandgap para una placa cuadrada son aún menores que los encontrados en placas con orificio, esto es debido a que en placas cuadradas es más posible encontrar bandgap negativos. Los tiempos de optimización se mantienen en el mismo orden de magnitud en ambos casos.

En la figuras 4.26, figuras 4.27 y 4.28 y figura 4.29 se presentan las topologías, diagramas de bandas y curvas de convergencia respectivamente, correspondientes a los mejores resultados de las tablas mostradas anteriormente, considerando todos los radios trabajados, para placas de 16 y 32 elementos por arista.

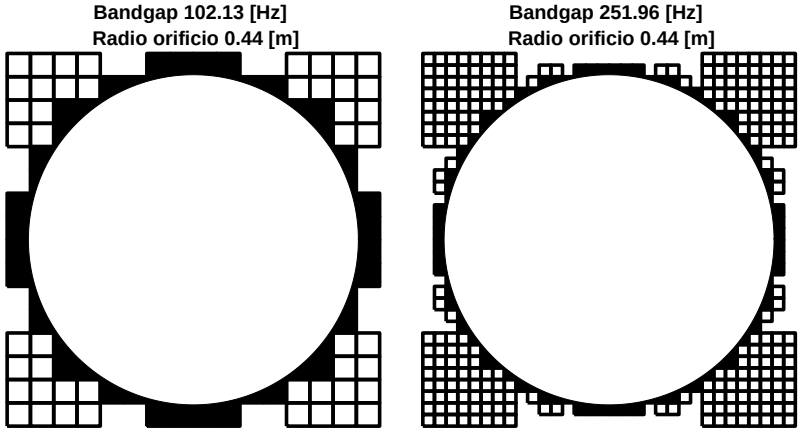


Figura 4.26: Topologías resultantes de optimización de placas cuadradas con orificios circulares, placas de 16x16 elementos (Izquierda) y de 32x32 elementos (Derecha). Elementos completamente negros representan los elementos fijos y por tanto, fuera del espacio de optimización del algoritmo.

La figura 4.26 muestra que para los dos tipos de placas usadas se llega a que el máximo valor del bandgap se encuentra en radios de 0.44 [m], donde además las topologías son muy similares, aunque no iguales, en ambos casos. A pesar de lo anterior, los valores de bandgaps son diferentes en cada caso, siendo mayor para placas de 32x32 elementos. Esta diferencia se debe a las diferencias menores en la topología, y a que la sección con forma circular tiene un espesor menor para la placa de 16x16 elementos. Los elementos pintados en negro son los elementos fijados, mientras que los demás elementos son los que deciden ser construidos luego de la optimización topológica.

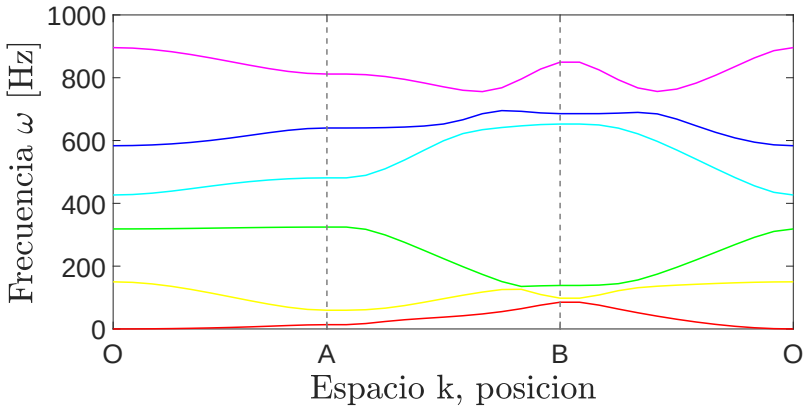


Figura 4.27: Diagrama de bandas resultante de optimización de placa cuadrada de 16 elementos por arista con orificio circular de 0.44 [m].

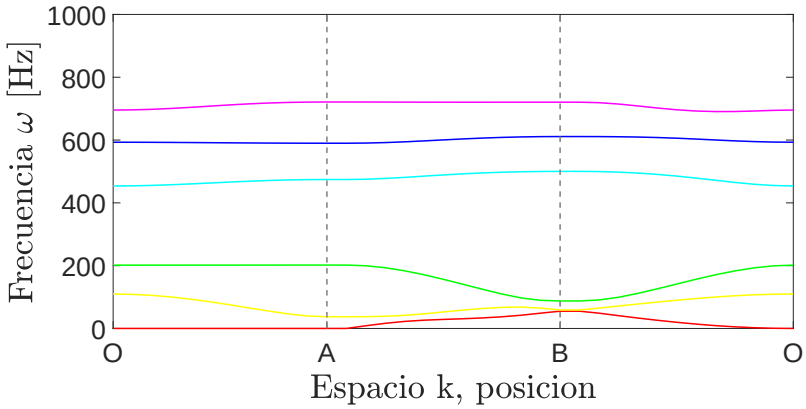
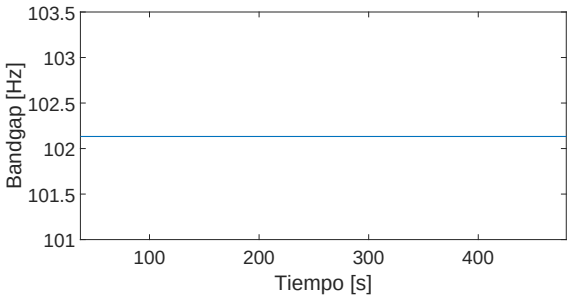
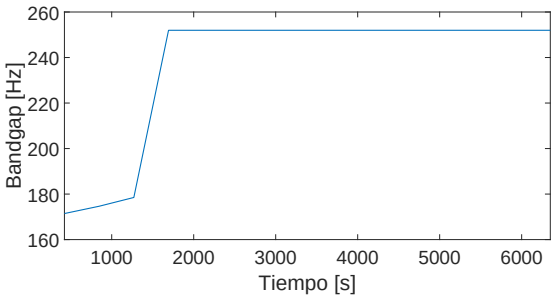


Figura 4.28: Diagrama de bandas resultante de optimización de placa cuadrada de 32 elementos por arista con orificio circular de 0.44 [m].

Las figuras 4.27 y 4.28 muestran los diagramas de bandas resultantes de la optimización, los cuales corresponden a las topologías mostradas en la figura 4.26, donde se ve claramente la diferencia en los valores de bandgaps en ambos casos. En ambas figuras es fácil ver que las frecuencias correspondientes la banda inferior tienen valores similares, mientras que las mayores diferencias se presentan en la banda superior, donde, para el caso de la placa de 32x32 elementos los valores tienden a ser más constantes, formándose incluso bandgaps mas pequeños entre cada frecuencia.



(a) Curva para placas de 16x16 elementos con orificio de 0.44 [m].



(b) Curva para placas de 32x32 elementos con orificio de 0.44 [m].

Figura 4.29: Curvas de convergencia para placas cuadrada de diferentes radios y diferente número de elementos totales.

La figura 4.29 muestra las curvas de convergencia para los casos analizados anteriormente, para la placa de 16x16 elementos se tiene que la curva es una recta constante, esto señala que desde la primera generación se llega al valor óptimo, esto es debido a que para el radio r_5 se tienen solo 9 grados de libertad, a diferencia de los 36 originales para el caso de una placa sin orificio, por lo tanto el algoritmo determina el valor optimo en la población inicial y solo sigue iterando hasta cumplir la condición de convergencia. La curva de convergencia de la placa de 32x32 elementos no presenta el comportamiento anterior, pero si converge al óptimo en un número bajo de iteraciones.

Capítulo 5

Conclusiones

Tanto FEA como IGA son capaces de producir resultados satisfactorios, con valores de frecuencias comparables entre si y a la bibliográfica consultada. La principal diferencia entre ambos métodos radica en que las matrices de masa y rigidez son de mayor tamaño en FEA, lo que repercute en el tiempo utilizado en el cálculo de frecuencias normales. Se cumple además que las contribuciones en el tiempo de ejecución de ambos algoritmos escalan diferente al momento de aumentar el número de elementos, aumentando siempre en menor medida para el caso de IGA. Además se tiene que FEA es independiente del mallado para placas de más elementos cuando se es comparado con IGA.

Los algotirmos MpGA y MBPSO son capaces de realizar optimizaciones de acuerdo a los parámetros impuestos con diferentes grados de éxito. El algoritmo MBPSO no solo es viable y es capaz de realizar optimizaciónes en placas cuadradas, sino que es el algoritmo preferido para este tipo de trabajos, por sobre MpGA, dado que este último es siempre mas lento, sin importar que tipo de placa o que algoritmo para modelarla se esté usando y entrega valores de bandgaps menores en promedio.

Para todos los casos de estudio en optimizaciones en placas cuadradas se obtienen bandgaps no positivos aun cuando el promedio lo es, es sugiere que los algoritmos no están ajenos a quedarse atrapados en óptimos locales. Contrario a esto, cuando se trabaja con placas con orificio, se obtiene que el algoritmo es mucho menos propenso al comportamiento anterior, esto es debido a que disminuye el espacio solución cuando se agregan restricciones adicionales en la geometría, y por tanto disminuye el número de configuraciones de placas posibles.

Se tiene una gran variabilidad en los resultados para todos los algoritmos de optimización usados, eso se explica en base a que cada uno de ellos tiene un componente aleatorio, reproducción por ranking, entrecruzamiento y mutación para el caso de MpGA y los cálculos de $v_i(t)$ y $x_i(t+1)$ para MBPSO. Debido a esto cada optimización es diferente a la anterior, sin importar si las poblaciones iniciales son puedan ser iguales.

Al realizar la optimización un gran número de veces se llega a la conclusión que la probabilidad de obtener el valor óptimo es bajo, pero la probabilidad de obtener un valor alto no lo es, por ejemplo la probabilidad acumulada evaluada en 40[Hz] es de 45.28 %. Dado que se busca estimar los valores de probabilidad para valores altos de fitness, se recomienda usar la

distribución Estable por sobre la KDE.

Observando las topologías resultantes de la optimización, estas presentan la misma forma base para todos los casos (salvo pequeñas variaciones), esto propone que los algoritmos de optimización se comportan de manera similar bajo las mismas condiciones, sin importar como se modele la placa o que tan refinada esté la malla.

Para placas cuadradas, la convergencia de los algoritmos no depende de la población inicial. Esto sugiere que, dada la cantidad de individuos iniciales, se tiene suficiente variabilidad para llevar a cabo cada optimización. Este comportamiento se ve aun más claro para el caso de optimización en placas con orificio, dado que las desviaciones estándar son menores y por tanto los resultados son mas homogéneos.

Dados los resultados referentes a bandgaps y tiempos de optimización en placas cuadradas, se obtiene que la combinación de IGA con MPBSO es la óptima para realizar este tipo de procedimientos, y se recomiendo usar para futuros trabajos.

A pesar de la tendencia creciente del bandgap conforme aumenta el tamaño del orificio, se obtiene que al momento de realizar la optimización topológica este comportamiento se pierde en el caso de radios pequeños. Para el caso de el último radio analizado en placas de 32 elementos por arista, este comportamiento se le atribuye a la falta de precisión encontrada para placas con espesores menores al tamaño de un elemento.

Existen radios para los cuales no se encuentran bandgaps positivos en ninguna de las 10 repeticiones de la optimización, esto indica que para esos radios no existen bandgaps. Por otra parte, se encuentra que el radio óptimo es el mismo para los dos tipos de mallas usadas, lo que señala que este es el radio óptimo del problema.

Comparando placas con y sin orificio con diferente número de elementos por arista se obtiene que las placas con orificio con 32 elementos por arista son las que arrojan los mejores resultados (promedios e individuales), también tienen tiempos de optimización menores en comparación a placas sin orificio.

Bibliografía

[1] M. P. Bendsøe and N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Comput. Methods Appl. Mech. Eng., 1988, doi: 10.1016/0045-7825(88)90086-2.

[2] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Eng., 2005, doi: 10.1016/j.cma.2004.10.008.

[3] H. D. Morgan, H. U. Levatti, J. Sienz, A. J. Gil, and D. C. Bould, GE Jet Engine Bracket Challenge: A Case Study in Sustainable Design, J. Innov. Impact, 2014.

[4] Y. Wang, Z. Wang, Z. Xia, and L. H. Poh, Structural design optimization using isogeometric analysis: A comprehensive review, CMES - Computer Modeling in Engineering and Sciences. 2018, doi: 10.31614/cmes.2018.04603.

[5] A. Dumont, Adjoint-Based Aerodynamic Shape Optimization Applied to Morphing Technology on a Regional Aircraft Wing, in Morphing Wing Technologies, Elsevier, 2018, pp. 145174.

[6] W. Zhang, J. Zhu, and T. Gao, Integrated Layout and Topology Optimization, in Topology Optimization in Engineering Structure Design, Elsevier, 2016, pp. 159215.

[7] Y.-L. Ma, J.-R. Tan, D.-L. Wang, and Z.-Z. Liu, Light-Weight Design Method for Force-Performance-Structure of Complex Structural Part Based Co-operative Optimization, Chinese J. Mech. Eng., vol. 31, no. 1, p. 42, Dec. 2018, doi: 10.1186/s10033-018-0236-3.

[8] E. Stein, History of the Finite Element Method Mathematics Meets Mechanics Part I: Engineering Developments, 2014.

[9] W. Ritz, Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik, J. fur die Reine und Angew. Math., 1909, doi: 10.1515/crll.1909.135.1.

[10] B.G. Galerkin,: Sterzhni i plastiny. Ryady v nekotorykh voprosakh uprugogo ravnovesiya sterzhnei i plastin (Rods and plates series occuring in some problems of elastic equilibrium of rods and plates), Vestnik Inzhenerov i Tekhnikov, Petrograd, Vol.19, pp. 897908 (1915) (in Russian) (English Translation: 63-18925, Clearinghouse Fed. Sci. Tech. Info. 1963).

[11] K.E. Buck, D.W. Scharpf, E. Schrem, E. Stein,: Einige allgemeine Programmsysteme für finite Elemente. In: Buck, K.E., Scharpf, D.W., Stein, E., Wunderlich, W. (eds.) Finite Elemente in der Statik, pp. 399454. Wilhelm Ernst & Sohn, Berlin (1973).

[12] A. S. Plouin and E. Balmes, Test validated model of plates with constrained viscoelastic materials, 1999.

[13] A. S. Plouin and E. Balmes, Steel/viscoelastic/steel sandwich shells computational

methods and experimental validations, Proc. Int. Modal Anal. Conf. - IMAC, 2000.

[14] G. Vermot Des Roches and E. Balmes, Understanding friction induced damping in bolted assemblies through explicit transient simulation, 2014.

[15] G. V. Roches, E. Balmes, R. Lemaire, and T. Pasquet, Design oriented simulation of contact-friction instabilities in application to realistic brake assemblies, 2010.

[16] A. Sternchüss and E. Balmès, Reduction of multistage rotor models using cyclic modeshapes, 2007, doi: 10.1115/GT2007-27974.

[17] E. Balmes and I. Bucher, Accounting for rotation in a multi-stage cyclo-symmetric model - A case study, 2010.

[18] G. Bhardwaj, I. V. Singh, and B. K. Mishra, Fatigue crack growth in functionally graded material using homogenized XIGA, Compos. Struct., 2015, doi: 10.1016/j.compstruct.2015.08.065.

[19] G. Bhardwaj, I. V. Singh, B. K. Mishra, and T. Q. Bui, Numerical simulation of functionally graded cracked plates using NURBS based XIGA under different loads and boundary conditions, Compos. Struct., 2015, doi: 10.1016/j.compstruct.2015.02.066.

[20] P. Antolin, A. Buffa, and L. Coradello, A hierarchical approach to the a posteriori error estimation of isogeometric Kirchhoff plates and KirchhoffLove shells, Comput. Methods Appl. Mech. Eng., 2020, doi: 10.1016/j.cma.2020.112919.

[21] A. Buffa and C. Giannelli, Adaptive isogeometric methods with hierarchical splines: Error estimator and convergence, Math. Model. Methods Appl. Sci., 2016, doi: 10.1142/S0218202516500019.

[22] L. Coradello, P. Antolin, R. Vázquez, and A. Buffa, Adaptive isogeometric analysis on two-dimensional trimmed domains based on a hierarchical approach, Comput. Methods Appl. Mech. Eng., 2020, doi: 10.1016/j.cma.2020.112925.

[23] J. Winiewski, Optimal design of reinforcing fibres in multilayer composites using genetic algorithms, Fibres Text. East. Eur., 2004.

[24] L. RatnaRaju and N. T.K., Design Optimization of Perforated Plate Heat Exchangers using Genetic Algorithm, Int. J. Adv. Mater. Manuf. Charact., 2013, doi: 10.11127/ijammc.2013.02.016.

[25] C. Lamini, S. Benhlima, and A. Elbekri, Genetic algorithm based approach for autonomous mobile robot path planning, 2018, doi: 10.1016/j.procs.2018.01.113.

[26] S. Lee, S. Soak, S. Oh, W. Pedrycz, and M. Jeon, Modified binary particle swarm optimization, Prog. Nat. Sci., 2008, doi: 10.1016/j.pnsc.2008.03.018.

[27] J. Kennedy and R. C. Eberhart, Discrete binary version of the particle swarm algorithm, 1997, doi: 10.1109/icsmc.1997.637339.

[28] J. Kennedy and R. Eberhart, Particle swarm optimization, 1995, doi: 10.4018/ijmfmp.2015010104.

[29] C. W. Chan, Modified Particle Swarm Optimization Algorithm for Multi-Objective Optimization Design of Hybrid Journal Bearings, J. Tribol., 2015, doi: 10.1115/1.4028606.

[30] C. Zhou, L. Gao, H. B. Gao, and K. Zan, Particle swarm optimization for simultaneous

optimization of design and machining tolerances, 2006, doi: 10.1007/11903697_110.

[31] M. M. Noor, K. Kadirgama, and M. M. Rahman, Particle swarm optimisation prediction model for surface roughness, Int. J. Phys. Sci., 2011, doi: 10.5897/IJPS10.263.

[32] A. Le van, Kirchhoff-Love Plate Theory, in Nonlinear Theory of Elastic Plates, Elsevier, 2017, pp. 83127.

[33] G. R. Liu and S. S. Quek, FEM for Plates and Shells, in The Finite Element Method, Elsevier, 2014, pp. 219247.

[34] V. P. Nguyen, C. Anitescu, S. P. A. Bordas, and T. Rabczuk, Isogeometric analysis: An overview and computer implementation aspects, Math. Comput. Simul., 2015, doi: 10.1016/j.matcom.2015.05.008.

[35] A. S. Phani, J. Woodhouse, and N. A. Fleck, Wave propagation in two-dimensional periodic lattices, J. Acoust. Soc. Am., 2006, doi: 10.1121/1.2179748.

[36] S. Mirjalili and A. Lewis, S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization, Swarm Evol. Comput., 2013, doi: 10.1016/j.swevo.2012.09.002.

[37] G. R. Liu and S. S. Quek, FEM for Two-Dimensional Solids, in The Finite Element Method, Elsevier, 2014, pp. 161217.

[38] D. F. Rogers, An introduction to NURBS: with historical perspective. 2001.

[39] J. M. Martínez-Duart, R. J. Martín-Palma, and F. Agulló-Rueda, Survey of Solid State Physics, in Nanotechnology for Microelectronics and Optoelectronics, Elsevier, 2006, pp. 2153.

[40] P. K. Misra, Basic Properties of Crystals, in Physics of Condensed Matter, Elsevier, 2012, pp. 135.

[41] P. K. Misra, Nearly Free Electron Model, in Physics of Condensed Matter, Elsevier, 2012, pp. 95130.

[42] P. K. Misra, Nearly Free Electron Model, in Physics of Condensed Matter, Elsevier, 2012, pp. 95130.

[43] G. Yi, Y. C. Shin, H. Yoon, S.-H. Jo, and B. D. Youn, Topology optimization for phononic band gap maximization considering a target driving frequency, JMST Adv., 2019, doi: 10.1007/s42791-019-00019-y.

[44] K. Gopalakrishnan, Particle Swarm Optimization in Civil Infrastructure Systems, in Metaheuristic Applications in Structures and Infrastructures, Elsevier, 2013, pp. 4976.

[45] G. C. Luh, C. Y. Lin, and Y. S. Lin, A binary particle swarm optimization for continuum structural topology optimization, 2011, doi: 10.1016/j.asoc.2010.11.013.

[46] A. K. B. Mohammad Zahid Rayaz Khan, Genetic Algorithm And Its Application In Mechanical Engineering, Int. J. Eng. Res. Technol., vol. 2, no. 5, 2013.

[47] E. Cantú-Paz, A Survey of Parallel Genetic Algorithms, Calc. Paralleles Reseaux Syst. Repart., 1998, doi: 10.1007/s11460-012-0192-0.

Anexos

Anexo A

Teoría de modelado de placas

A.1. FEA

A.1.1. Funciones de forma

Se le llama funciones de forma a la base de funciones **N** que interpolan la solución obtenida en los nodos para el modelo de elementos finitos. Estas funciones corresponden a una base polinomial y se expresan en función de un sistema de coordenadas local, por ende, ésta definición puede ser usada por cualquier elemento que se quiera definir, sin importar su tamaño ni su posición dentro del sistema de coordenadas global [37, 33].

Las funciones de forma cumplen las siguientes propiedades:

- Son linealmente independientes.
- Son partición de la unidad, o sea se cumple que:

$$\sum_i N_i(\mathbf{x}) = 1, \quad \forall \mathbf{x} \tag{A.1}$$

- Propiedad de delta de Dirac:

$$N_i(\mathbf{x_j}) = \delta_{i,j} \tag{A.2}$$

- Propiedad de consistencia: Sea una función forma N_i, si ésta está construida en base a polinomios de hasta orden k, entonces la función de forma es C^k, es decir es continuamente derivable k veces.
- Reproducción de campos lineales: Para una propiedad $f(\mathbf{x})$, la cual puede escribirse como combinación de polinomios de hasta orden k sobre la cantidad total de nodos de

una malla (n_d), se tiene que:

$$f(\mathbf{x}) = \sum_{i}^{n_d} p_i(\mathbf{x})\beta_i \tag{A.3}$$

Luego, si la base de polinomios son monomios lineales, entonces se tiene que:

$$f^h(\mathbf{x}) = \sum_{i}^{n_d} N_i(\mathbf{x})\mathbf{x_i} \tag{A.4}$$

Donde $\mathbf{x_i}$ son los valores del campo lineal en los nodos y $f^h(\mathbf{x})$ es la aproximación de la propiedad según elementos finitos.

Para el caso 2D, las funciones de forma más utilizadas corresponden a las bilineales, representadas en su forma general de la siguiente manera:

$$N(\xi, \eta) = A + B\xi + C\eta + D\xi\eta \tag{A.5}$$

A continuación, se muestra un ejemplo de como se relacionan los sistemas de coordenadas local y global en el caso 2D.

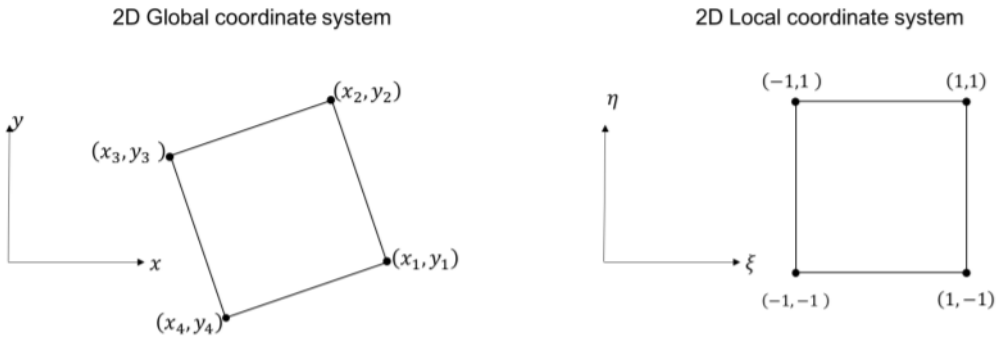


Figura A.1: Relación entre sistemas coordenados local y global.

Para el caso específico de elementos cuadrados, en particular para el ejemplo anterior, se tienen las siguientes expresiones para las funciones de forma:

$$\begin{aligned} N_1(\xi, \eta) &= \frac{1}{4}(1-\xi)(1-\eta) \\ N_2(\xi, \eta) &= \frac{1}{4}(1+\xi)(1-\eta) \\ N_3(\xi, \eta) &= \frac{1}{4}(1+\xi)(1+\eta) \\ N_4(\xi, \eta) &= \frac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{A.6}$$

A.2. IGA

A.2.1. Non Uniform Rational B-Spline NURBS

Sea $\Xi = \xi_1, \ldots, \xi_{n+p+1}$ una secuencia no decreciente de números reales positivos, es decir $0 < \xi_i \leq \xi_{i+1}$, $\forall i = 1, \ldots, n+p$. El conjunto Ξ definido por ξ_i es llamado vector *knot* y *knot* respectivamente, donde p es el orden del polinomio que compone la NURBS y n es el número de funciones de la base que componen la NURBS. Si los *knot* están equiespaciados, entonces se dice que la base es uniforme, de lo contrario se dice que la base es no uniforme. El intervalo abierto $[u_i, u_{i+1})$ se define como el i-ésimo *knot span*, los *knot* pueden ser considerados como los puntos que subdividen un intervalo cerrado en *knot spans* [34, 38].

Una NURBS de orden p es definida de la siguiente manera:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} R_{i,p}\mathbf{T_i} \tag{A.7}$$

donde $\mathbf{T_i}$ son la posición de un set de $i = 1, 2 \ldots n$ puntos de control, mientras que $R_{i,p}$ son las funciones definidas por NURBS, las cuales se definen como sigue:

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi)w_i}{\sum_{j=1}^{n} N_{j,p}(\xi)w_j} \tag{A.8}$$

donde w_i son los pesos asociados a cada $R_{i,p}(\xi)$ y $N_{i,p}(\xi)$ son B-spline (o basis spline) de orden p, las cuales son una base de funciones definida de forma recursiva por:

$$N_{i,0}(\xi) = \begin{cases} 1 & si \quad \xi_i \leq \xi \leq \xi_{i+1} \\ \\ 0 & si \qquad\quad no \end{cases} \tag{A.9}$$

De esta manera se define $N_{i,p}(\xi)$ para los otros casos:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \quad p = 1, 2, 3 \ldots \tag{A.10}$$

Análogamente a lo anterior, se puede definir una superficie NURBS de orden p en la dirección ξ y de orden q en la dirección η

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q}\mathbf{T_{i,j}} \tag{A.11}$$

donde $\mathbf{T_{i,j}}$ es una malla de n x m puntos de control y $R_{i,j}^{p,q}$ se define como sigue:

$$R_{i,j}^{p,q}(\xi, \eta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta)w_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi)M_{j,q}(\eta)w_{i,j}} \tag{A.12}$$

donde $N_{i,p}(\xi)$ y $M_{j,q}(\eta)$ son B-splines definidas sobre los *knots* Ξ_1 y Ξ_2, respectivamente.

La primera derivada de $R_{i,j}^{p,q}(\xi, \eta)$ con respecto a cualquiera de sus dos coordenadas paramétricas se obtiene fácilmente al aplicar la regla del cuociente para derivadas, por ejemplo, para la coordenada ξ se tiene lo siguiente:

$$\frac{R_{i,j}^{p,q}}{\partial\xi} = \frac{\frac{\partial N_{i,p}(\xi)}{\partial\xi} M_{j,q}(\eta) W(\xi,\eta) - \frac{\partial W(\xi,\eta)}{\xi} N_{i,p}(\xi) M_{j,q}(\eta) w_{i,j}}{(W(\xi,\eta))^2} \tag{A.13}$$

$$W(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi) M_{j,q}(\eta) w_{i,j} \tag{A.14}$$

$$\frac{\partial W(\xi,\eta)}{\partial\xi} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial N_{i,p}(\xi)}{\partial xi} M_{j,q} w_{i,j} \tag{A.15}$$

Las derivadas de orden superior se pueden obtener de la misma manera.

Las propiedades mas importantes de las NURBS pueden ser resumidas en los siguiente puntos:

- Partición de la unidad, $\forall\xi, \sum_{i=1}^{n} R_{i,p}(\xi) = 1$
- No negatividad, $\forall\xi, R_{i,p}(\xi) \geq 0$
- Funciones base de orden p son p-m_i veces continuamente diferenciable sobre un *knot* ξ_i, donde m_i es la multiplicidad del valor ξ_i en el vector *knot* correspondiente.
- Soporte local, es decir el soporte de $R_{i,j}^{p,q}$ es compacto y contenido en el intervalo $[\xi_i, \xi_{i+p+1}]$.

A.3. Teorema de Bloch

El teorema de Bloch [39] enuncia que todo campo vectorial generado a partir de una celda periódica hereda la periodicidad de la celda.

Una onda típica, dentro de una estructura cristalina, es llamada onda de Bloch y es matemáticamente representada de la siguiente manera:

$$\psi(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}}\mathbf{u}(\mathbf{r}) \tag{A.16}$$

donde $\psi(\mathbf{r})$ es la onda de Bloch, $\mathbf{r}$ es la posición, $\mathbf{u}$ es una función con la misma periodicidad de la celda y $\mathbf{k}$ es el vector de onda.

Para aplicar el teorema de Bloch al cálculo de desplazamientos y frecuencias normales en una placa se deben hacer las siguientes definiciones.

Sean $\mathbf{r_j}$ los puntos pertenecientes a una celda unitaria, $\mathbf{q}(\mathbf{r_j})$ los desplazamientos de los puntos de la celda en la celda de referencia. Si se considera que $\mathbf{q}(\mathbf{r_j})$ admite soluciones de onda plana, entonces:

$$q(\mathbf{r_j}) = q_j e^{i\omega t - \mathbf{k}\cdot\mathbf{r_j}} \tag{A.17}$$

donde q_j es una amplitud, ω es la frecuencia de vibración y $\mathbf{k}$ es el vector de onda.

Sean (n_1, n_2) un par de números enteros que definen una traslación $\mathbf{R_n}$ en los sentidos $\mathbf{e_1}$ y $\mathbf{e_2}$ respectivamente, se tiene un nuevo punto en la celda $\mathbf{r} = \mathbf{r_j} + \mathbf{R_n} = \mathbf{r_j} + n_1\mathbf{e_1} + n_2\mathbf{e_2}$.

Usando el teorema de Bloch, el desplazamiento del j-ésimo punto en una celda dada por los puntos (n_1, n_2) se escribe de la siguiente manera:

$$q(\mathbf{r}) = q(\mathbf{r_j})e^{\mathbf{k}\cdot(\mathbf{r}-\mathbf{r_j})} = q(\mathbf{r_j})e^{(k_1 n_1 + k_2 n_2)} \tag{A.18}$$

donde $k_1 = \mathbf{k} \cdot \mathbf{e_1}$ y $k_2 = \mathbf{k} \cdot \mathbf{e_2}$. Considerando además que $k_1 = \delta_1 + i\epsilon_1$ y $k_2 = \delta_2 + i\epsilon_2$, donde a las partes real y parte imaginaria se llaman contantes de atenuación y de fase, respectivamente. La parte real es una medida de la atenuación que la onda presenta al pasar de una celda a otra. Para ondas sin atenuación se tiene que $\delta_1 = \delta_2 = 0$, luego $k_1 = i\epsilon_1$ y $k_2 = i\epsilon_2$.

Usando el teorema de Bloch se puede caracterizar la propagación de las ondas a través de una red periódica solo considerando y calculando las ondas en una celda unitaria.

Trabajando en el espacio recíproco, considerando que la red recíproca también es periódica y a las simetrías de una red en específico, se puede restringir el cálculo a la primera zona irreductible de Brillouin.

A.4. Primera Zona de Brillouin

La primera zona de Brillouin es el volumen más pequeño totalmente encerrado por planos que son bisectores perpendiculares de vectores del espacio recíproco dibujados desde el origen de su sistema coordenado [40, 41]. A priori, existen infinitas zonas de Brillouin, donde cada una no tiene intersección con las demás. En la figura A.2 se muestran las primeras 4 zonas de Brillouin para una celda cuadrada.

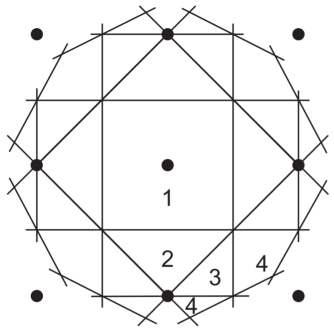


Figura A.2: Primeras 4 zonas de Brillouin [42].

En base a lo anterior, se define la primera zona irreductible de Brillouin la cual corresponde a la primera zona de Brillouin luego de aplicarse las simetrías propias de la geometría de la red [43]. Al aplicarle estas relaciones de simetría se restringen los valores que puede tomar el vector de onda $\mathbf{k}$, luego sus proyecciones k_1 y k_2 se definen según el camino mostrado en la figura A.3.

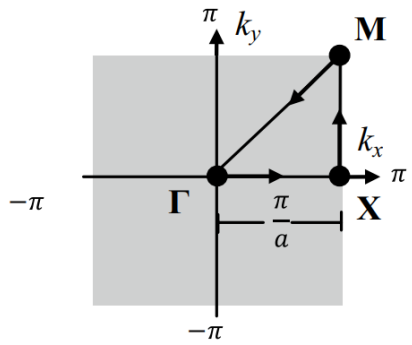


Figura A.3: Recorrido a seguir dentro de la primera zona irreductible de Brillouin [43].

Anexo B

Algoritmos de optimización

B.1. Particle Swarm Optimization

B.1.1. Particle Swarm Optimization (PSO)

PSO es un algoritmo evolutivo inspirado en el comportamiento de bandadas de aves. Se usa un conjunto de partículas (aves), las cuales vuelan o se mueven por el espacio en busca de la solución óptima para el problema [44].

Cada partícula usa como variables tanto su posición y velocidad en el instante actual como la distancia al mejor valor de la partícula hasta instante actual (p_{best}) y la mejor partícula, considerando todos los tiempos hasta el momento de la evaluación (g_{best}). La medida con que se comparan las partículas es la evaluación de la función objetivo del problema en un instante de tiempo (valor llamado fitness), lo anterior se define de la siguiente manera:

$$\begin{aligned} v_i^{t+1} &= wv_i^t + c_1u(p_{best,i} - x_i^t) + c_2u(g_{best} - x_i^t) \\ x_i^{t+1} &= x_i^t + v_i^{t+1} \\ Fitness_i^{t+1} &= F(x_i^{t+1}) \end{aligned} \tag{B.1}$$

donde v_i^{t+1} y x_i^{t+1} son la velocidad y la posición de la partícula en el instante $t+1$ y de la partícula i, w es un valor que otorga un peso a la velocidad del instante actual, u es una variable aleatoria que sigue una distribución uniforme en el intervalo $[0, 1]$, c_1 y c_2 son términos llamados "de aceleración", los cuales otorgan un peso a sus respectivos sumandos, $p_{best,i}$ es el mejor valor que ha tenido la partícula i-esima, mientras que $p_{best,i}$ es el mejor valor que se ha tenido considerando todas las partículas de conjunto. $F(x)$ es la función objetivo, la cual depende completamente del problema que se quiera resolver.

El algoritmo comienza con una distribución aleatoria de las partículas en un espacio continuo, para luego ir calculando las velocidades y posiciones en las iteraciones futuras para cada partícula.

Para ayudar a la convergencia del algoritmo, se establecen límites para los valores que pueden tomar tanto la posición como la velocidad de cada partícula, los cuales son $[x_{min}, x_{max}]$ y $[v_{min}, v_{max}]$, respectivamente (valores previamente definidos en función del problema), im-

poniendo los siguientes límites:

$$v_i^{t+1} = \begin{cases} v_{max} & si \quad v_i^{t+1} > v_{max} \\ v_{min} & si \quad v_i^{t+1} < v_{min} \end{cases}$$
$$x_i^{t+1} = \begin{cases} x_{max} & si \quad x_i^{t+1} > x_{max} \\ x_{min} & si \quad x_i^{t+1} < x_{min} \end{cases} \tag{B.2}$$

El algoritmo se detiene cuando se cumple la condición mostrada en la ecuación B.8, misma condición utilizada como criterio de término para cada población de un algoritmo genético.

B.1.2. Binary Particle Swarm Optimization (BPSO)

BPSO se usa cuando se quiere trabajar en un espacio binario de para las variables que definen el problema o si se quiere transformar las variables continuas a variables binarias.

Dada la naturaleza de las variables, el espacio solución es un hipercubo cuyas aristas se ubican en las posiciones 0 o 1 para cada eje. Gracias a esto se deben hacer modificaciones a las expresiones de velocidad y posición definidas en la ecuación B.1. Para lograr la modificación se desarrollaron diferentes estrategias o métodos [45], las cuales se detallan a continuación:

Método 1

El primer método consiste en definirse una variable aleatoria v_i que siga una distribución uniforme en el intervalo $[0, 1]$ y una variable a que parte en el valor 0.5 y que vaya disminuyendo de forma uniforme su valor hasta alcanzar 0.33. Con estas variables se redefine como cambia la posición de cada partícula en cada iteración, lo que se muestra en la ecuación B.3.

$$x_i^{t+1} = \begin{cases} x_i^t & si \quad 0 < v_i \leq a \\ p_{best} & si \quad a < v_i \leq 0.5(1+a) \\ g_{best} & si \quad 0.5(1+a) < v_i \leq 1 \end{cases} \tag{B.3}$$

Método 2

Este método es el primero en ser propuesto y consiste en aplicarle a la velocidad una función denominada de transferencia, definida en el intervalo continuo $[0, 1]$, de la siguiente manera:

$$T(v_i^k(t)) = \frac{1}{1 + e^{-v_i^k(t)}} \tag{B.4}$$

donde $v_i^k(t)$ corresponde a la velocidad de la partícula i en el instante de tiempo t y en la componente k del espacio solución.

$T(v_i^k(t))$ se interpreta como una probabilidad y sirve como umbral para el cálculo de la posición en la iteración siguiente.

$$x_i^k(t+1) = \begin{cases} 0 & si \quad u < T(v_i^k(t)) \\ 1 & si \quad u \geq T(v_i^k(t)) \end{cases} \tag{B.5}$$

donde u es una variable aleatoria que sigue una distribución uniforme en el intervalo $[0, 1]$. Para terminar, la velocidad del intervalo siguiente $v_i^k(t+1)$ se calcula según el algoritmo PSO original, es decir, según la primera expresión de la ecuación B.1.

Método 3

El último método a tratar es el más reciente, el cual consiste primeramente en cambiar la función de transferencia a usar, usando la ecuación B.6. La posición en el paso siguiente del algoritmo se modifica según la ecuación B.7.

$$L(v_i^k(t)) = \frac{v_i^k(t) - R_{min}}{R_{max} - R_{min}} \tag{B.6}$$

$$x_i^k(t+1) = \begin{cases} 0 & si \quad u < L(v_i^k(t)) \\ 1 & si \quad u \geq L(v_i^k(t)) \end{cases} \tag{B.7}$$

donde $L(v_i^k(t))$ es la nueva función que transforma la velocidad en una probabilidad y $[R_{min}, R_{max}]$ son valores previamente definidos que aseguran que $L(v)$ esté en el rango de valores deseado, u es la variable aleatoria que sigue una distribución uniforme en el intervalo $[0, 1]$ y $v_i^k(t+1)$ se calcula según el algoritmo PSO original.

Todos lo métodos mostrados anteriormente tienen la tendencia de quedarse atrapados en algún mínimo local del problema, por lo que se busca una nueva formulación o modificación de las existentes que resuelva esta problemática.

B.2. Algoritmos genéticos

En esta sección se muestran los conceptos más importantes relacionados a los algoritmos genéticos o GA (Genetic Algorithms), esta sección se basa principalmente en [46, 47].

Los algoritmos genéticos son un tipo de algoritmo de optimización que está basado en los principios de la evolución natural de los seres vivos. Estos algoritmos cuentan una una población de un número determinado de individuos, dónde cada uno posee un set de características que los definen. Estas características o genes son los usados para determinar el valor de cada individuo con respecto al problema que se quiere representar y optimizar, conociendo este valor con el nombre de fitness.

La características reales de cada individuo son codificadas en los genes tal como sucede en moléculas de ADN. Por esta razón, los individuos cuentan con un análogo a los conceptos de genotipo y fenotipo, donde genotipo es la información genética y fenotipo es como el genotipo se expresa de una forma observable.

Durante el proceso de optimización, los algoritmos genéticos imitan procesos propios de la selección natural. Se cuenta con una regla o método de reproducción donde la población actual (padres) crean una población nueva (hijos), la cual tiene mayor probabilidad de tener un individuo más apto o capacitado, en el sentido del problema que se quiere optimizar (tener mejor fitness). Luego de generar una nueva población, se procede a realizar otros dos procesos, denominados entrecruzamiento y mutación, respectivamente, donde el primero consiste en

mezclar el material genético entre dos individuos, y el segundo consiste en cambiar el material genético de un individuo de forma aleatoria. El algoritmo termina cuando se alcanza el valor óptimo (en caso de ser conocido) o si, luego de alcanzar un número de generaciones predeterminado n_g, los mejores individuos separados por este número de generaciones son menores a un parámetro ϵ_1, lo que se resumen en la siguiente condición:

$$Si \quad \|Fitness_i - Fitness_{i-n_g}\| < \epsilon_1, \quad i > n_g \Rightarrow Parar \tag{B.8}$$

B.2.1. Reproducción

El método de reproducción que se describe a continuación es el denominado reproducción por ranking o selección por ranking, que consiste en establecer un ranking en función del fitness entre todos los individuos de una población, asignándole una probabilidad a cada uno, donde esta representa la probabilidad de ser escogido para formar la siguiente población.

La distribución usada en este trabajo es la denominada geométrica normalizada, la cual se define en función de la distribución geométrica usual $P(n)$ y su distribución acumulada $D(n)$, ambas en función de una probabilidad p conocida.

$$P(n) = p(1-p)^n \tag{B.9}$$

$$D(n) = \sum_{k=0}^{n} P(k) = 1 - (1-p)^n \tag{B.10}$$

La distribución geométrica normalizada $P'(n)$ se define en función de una probabilidad normalizada p', la cual se define en función de la distribución geométrica y su acumulada antes calculadas.

$$p' = \frac{P(n)}{D(n)} = \frac{p}{1-(1-p)^n} \tag{B.11}$$

$$P'(n) = p'(1-p)^n \tag{B.12}$$

Los individuos de la población actual se ordenan de acuerdo al valor de su fitness y a cada uno se le asigna un valor de probabilidad de acuerdo a la distribución de probabilidad P', donde el valor de n cambia de acuerdo a su posición dentro del ranking. Posteriormente, un individuo es elegido a formar parte de la siguiente población si su valor asignado de probabilidad es mayor a un cierto valor generado aleatoriamente (partiendo por el individuo de mayor ranking). Este proceso se repite hasta que se hayan generado todos los individuos para la nueva generación.

B.2.2. Entrecruzamiento

Al igual que con los seres vivos, el entrecruzamiento es un proceso de combinación de genes entre dos individuos (padres), donde en este caso, estos padres generan dos individuos (hijos). Existen diferentes tipos de entrecruzamiento, particularmente en este trabajo se usa el entrecruzamiento en un punto, el cual consiste en elegir un mismo segmento de cada padres, uniéndolo con el complemento del otro padre, por ejemplo, si se tienen los individuos definidos por $\{a, a, a, a, a\}$ y otro por $\{b, b, b, b, b\}$, haciendo entrecruzamiento a partir del gen número 3 se obtienen lo siguiente: $\{a, a, a, b, b\}$ y $\{b, b, b, a, a\}$.

B.2.3. Mutación

El proceso de mutación corresponde a la modificación de manera aleatoria de un o una serie de genes de cada individuo. Este proceso ocurre con una probabilidad predefinida en el problema, donde el proceso de mutación se realiza si un número aleatorio es mayor que la probabilidad de mutación establecida.

Anexo C

Funciones de distribución

Esta sección detalla lo básico de las funciones de distribución usadas en el ajuste de un histograma.

C.1. Distribución Estable

una distribución se dice estable si es la combinación lineal de dos o mas variables independientes que difieren solo en un factor de escala o parámetro de localización.

Una función estable es definida por 4 parámetros, estos son α, β, γ y δ, donde los primeros dos son parámetros de de forma, el tercero es un parámetro de forma y el cuarto es un parámetro de localización.

Si una variable aleatoria X sigue una distribución estable entonces su función de densidad de probabilidad está definida por la ecuación C.1, la cual a su vez está definida por su función característica, ver ecuación C.2, dado que no existe una forma analítica que sea general.

$$f(x) = \frac{1}{2\pi}\int_{\infty}^{\infty} \varphi(t)e^{-ixt}dt \tag{C.1}$$

$$\varphi(t;\alpha,\beta,\gamma,\delta) = \begin{cases} exp(-\gamma^{\alpha} \mid t \mid^{\alpha} \Big[1 + i\beta sign(t) tan(\frac{\pi\alpha}{2}((\gamma \mid t \mid)^{1-\alpha} - 1))\Big] + i\delta t) & para \quad a \neq 1 \\ exp(-\gamma \mid t \mid \Big[1 + i\beta sign(t)\frac{2}{\pi} ln(\gamma \mid t \mid)\Big] + i\delta t) & para \quad a = 1 \end{cases} \tag{C.2}$$

C.2. KDE

Estimación de densidad por Kernel o KDE por sus siglas en inglés es un método de estimación de una densidad de probabilidad de una variable aleatoria basado en kernels y pesos.

Sea un conjunto de muestras independientes e idénticamente distribuidas X_n, tomadas de una función de distribución desconocida f. Se cumple que su estimación mediante el método KDE es el que se muestra en la ecuación C.3.

$$f_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{C.3}$$

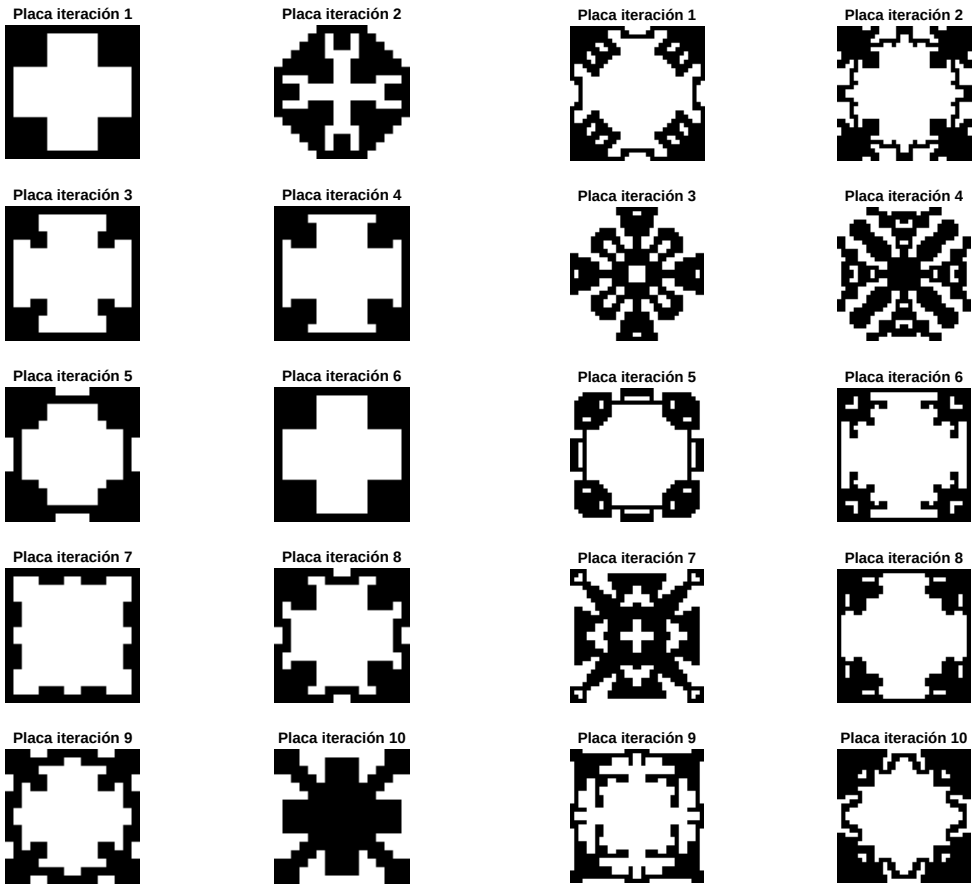
Donde K es el kernel o distribución usado para la estimación, h es el tamaño del parámetro llamado ancho de banda.

Anexo D

Resultados optimización en placas cuadradas

D.1. Placas resultantes

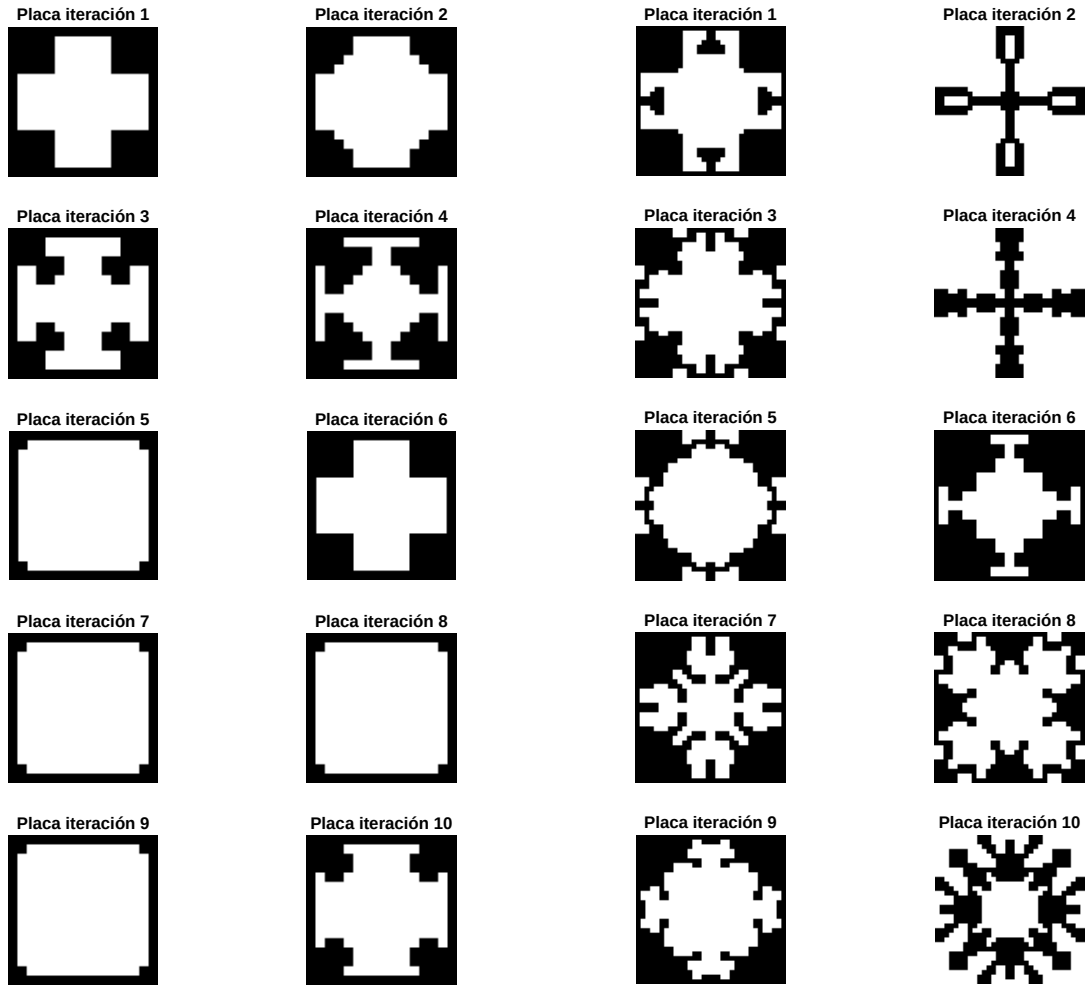
En la figura D.1 se muestran todas las placas resultantes de la optimización usando el modelo FEA para placas de 16 y 32 elementos por arista, optimizadas con MBPSO.



(a) Placas de $n = 16$. (b) Placas de $n = 32$.

Figura D.1: Placas resultantes de 10 optimizaciones, usando el algotirmo MBPSO y modeladas con FEA.

En la figura D.2 se muestran todas las placas resultantes de la optimización usando el modelo FEA para placas de 16 y 32 elementos por arista, optimizadas con el algoritmo MpGA.

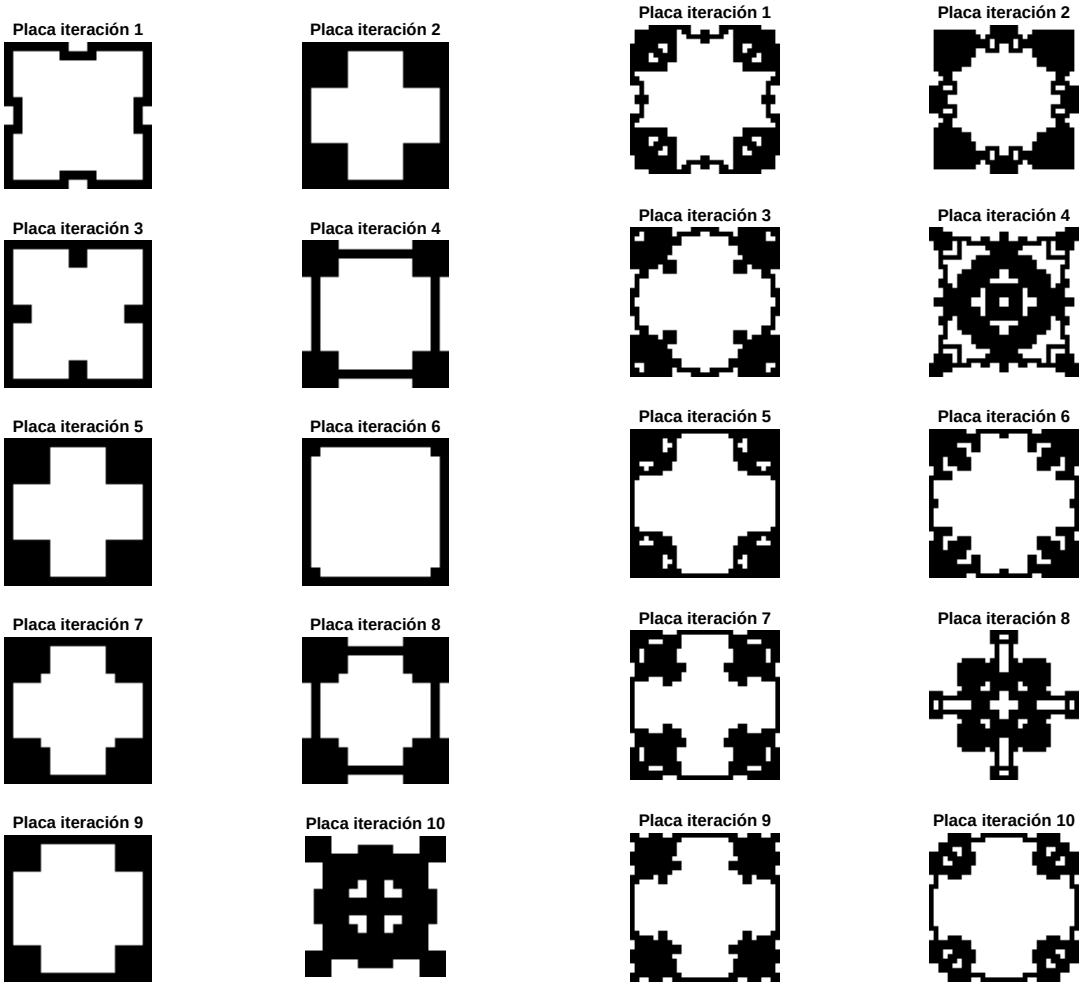


(a) Placas de $n = 16$.

(b) Placas de $n = 32$.

Figura D.2: Placas resultantes de 10 optimizaciones, usando el algotirmo MpGA y modeladas con FEA.

En la figura D.3 se muestran todas las placas resultantes de la optimización usando el modelo IGA para placas de 16 y 32 elementos por arista, optimizadas con el algoritmo MBPSO.

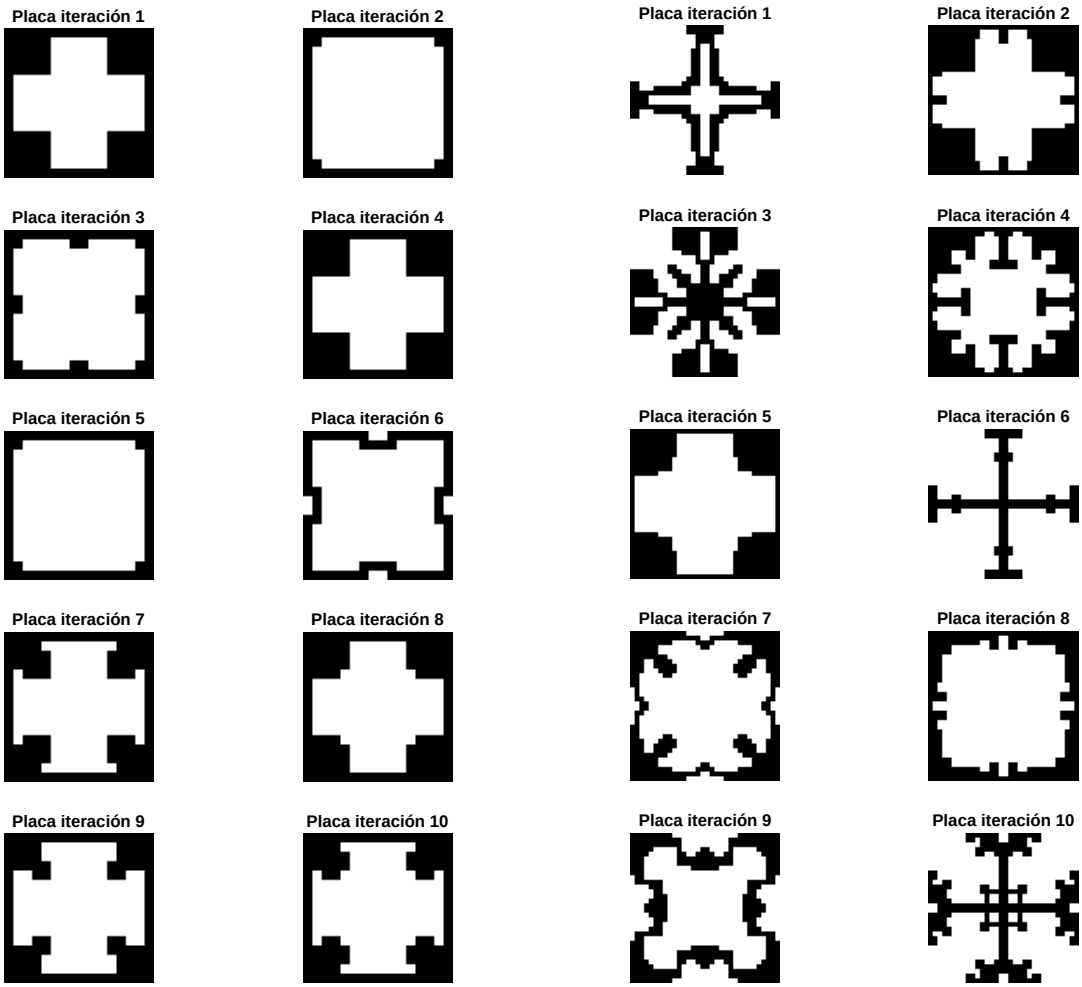


(a) Placas de $n = 16$.

(b) Placas de $n = 32$.

Figura D.3: Placas resultantes de 10 optimizaciones, usando el algotirmo MBPSO y modeladas con IGA.

En la figura D.4 se muestran todas las placas resultantes de la optimización usando el modelo IGA para placas de 16 y 32 elementos por arista, optimizadas con el algoritmo MpGA.



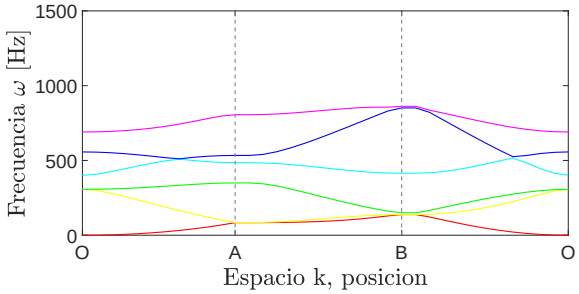
(a) Placas de $n = 16$.

(b) Placas de $n = 32$.

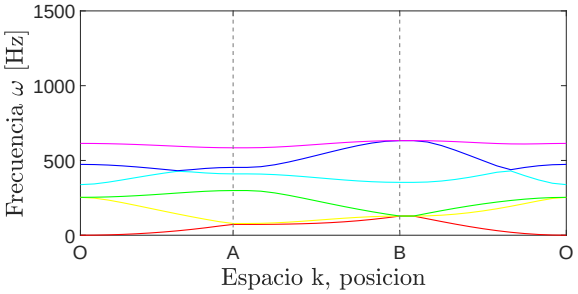
Figura D.4: Placas resultantes de 10 optimizaciones, usando el algotirmo MpGA y modeladas con IGA.

D.2. Diagramas de Bandas

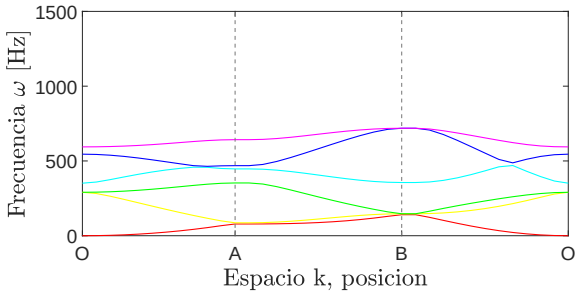
En las figuras D.5 y D.6 se presentan los diagramas de bandas para cada una de las placas resultantes modeladas con FEA optimizadas con MBPSO para $n = 16$ y $n = 32$ respectivamente.



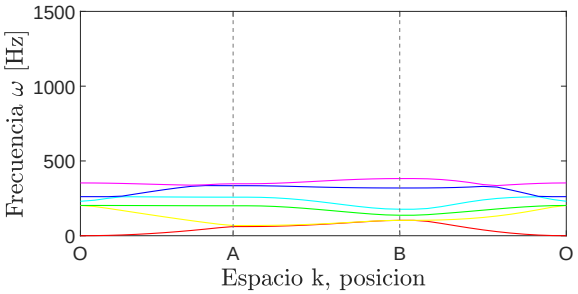
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.

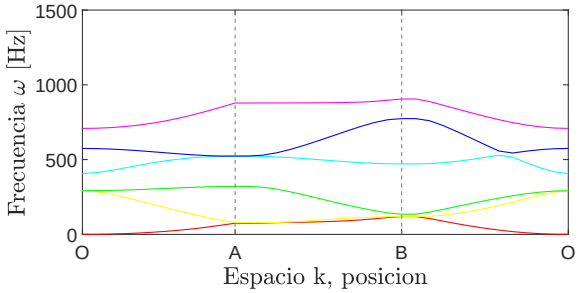


(c) Tercero ejemplo de diagrama de bandas.

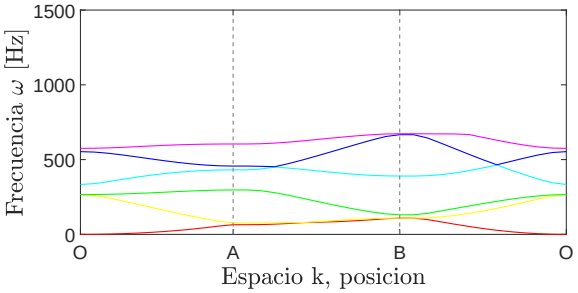


(d) Cuarto ejemplo de diagrama de bandas.

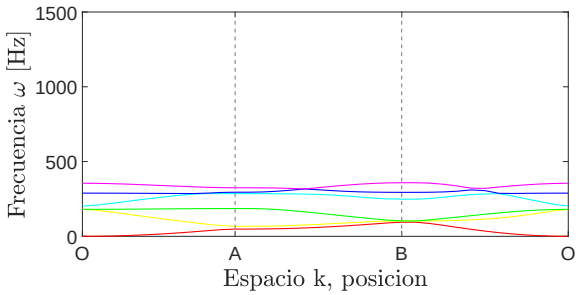
Figura D.5: Diagramas de bandas para placas modeladas con FEA y optimizadas con MBPSO para $n = 16$.



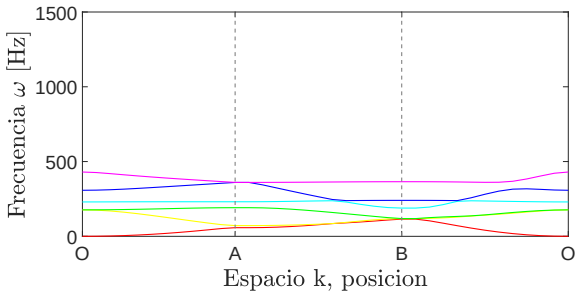
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.



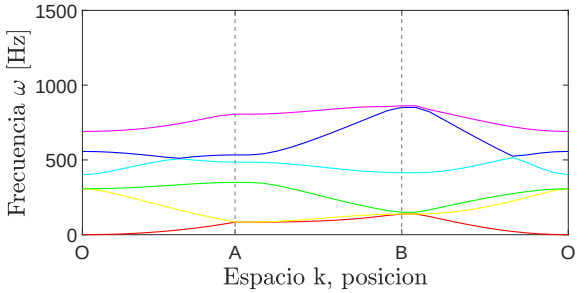
(c) Tercer ejemplo de diagrama de bandas.



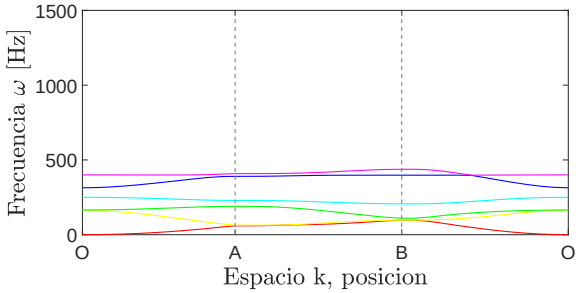
(d) Cuarto ejemplo de diagrama de bandas.

Figura D.6: Diagramas de bandas para placas modeladas con FEA y optimizadas con MBPSO para $n = 32$.

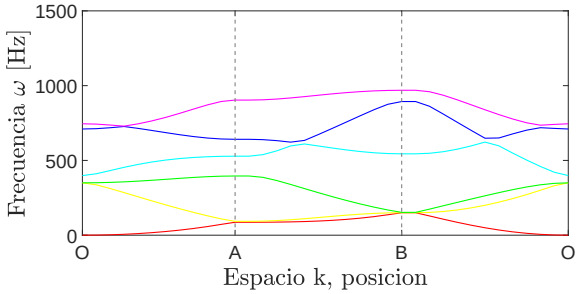
En las figuras D.7 y D.8 se muestran los diagramas de bandas para las placas resultantes modeladas con FEA, optimizadas con MpGA para $n = 16$ y $n = 32$ respectivamente.



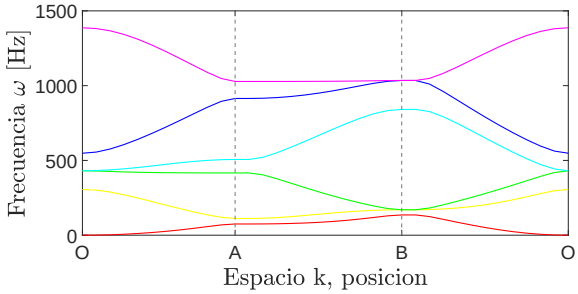
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.

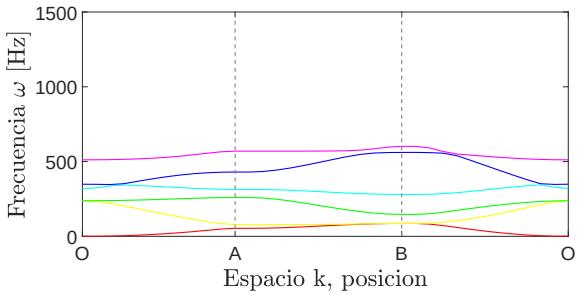


(c) Tercero ejemplo de diagrama de bandas.

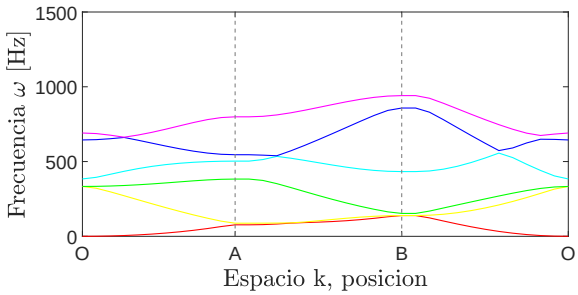


(d) Cuarto ejemplo de diagrama de bandas.

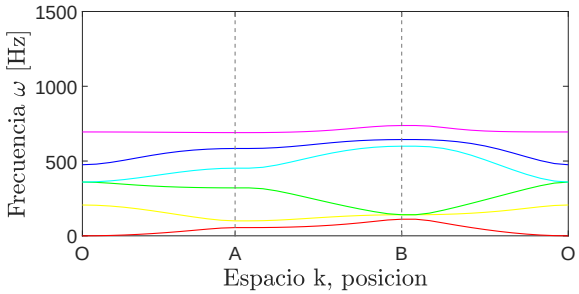
Figura D.7: Diagramas de bandas para placas modeladas con FEA y optimizadas con MBPSO para $n = 16$.



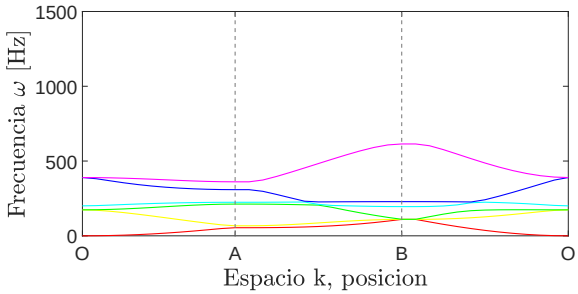
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.



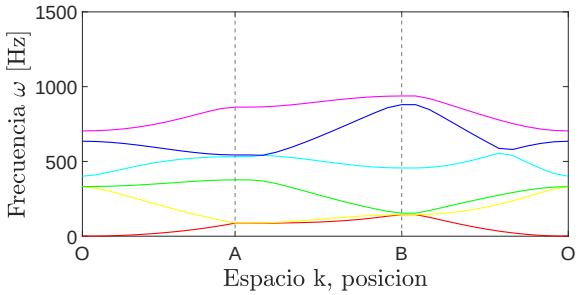
(c) Tercero ejemplo de diagrama de bandas.



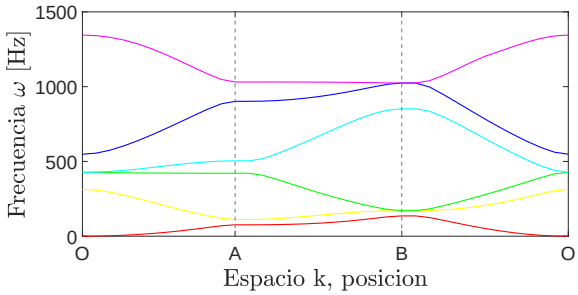
(d) Cuarto ejemplo de diagrama de bandas.

Figura D.8: Diagramas de bandas para placas modeladas con FEA y optimizadas con MpGA para $n = 32$.

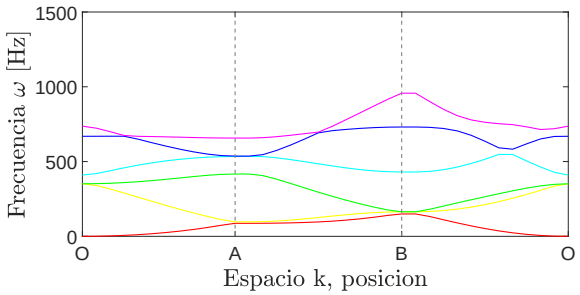
En las figuras D.9 y D.10 se muestran los diagramas de bandas para las placas resultantes modeladas con IGA, optimizadas con BPSO para $n = 16$ y $n = 32$ respectivamente.



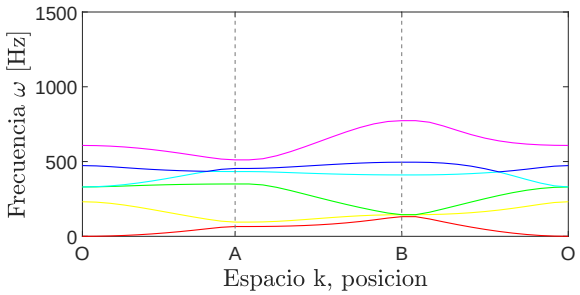
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.

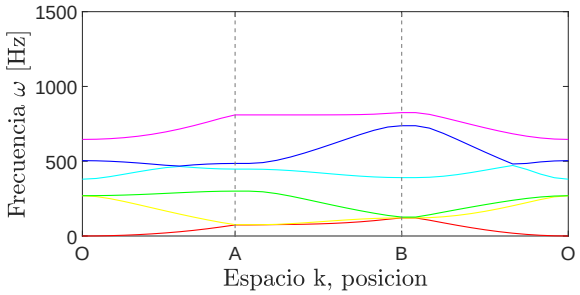


(c) Tercer ejemplo de diagrama de bandas.

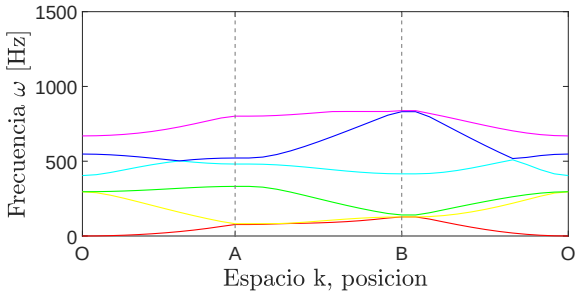


(d) Cuarto ejemplo de diagrama de bandas.

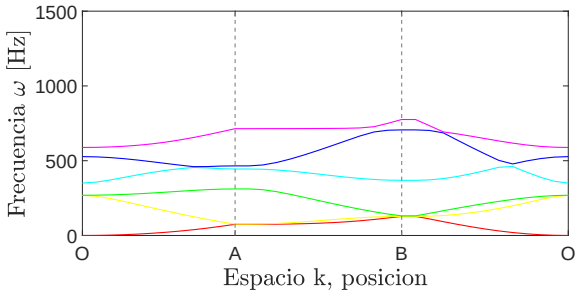
Figura D.9: Diagramas de bandas para placas modeladas con IGA y optimizadas con MBPSO para $n = 16$.



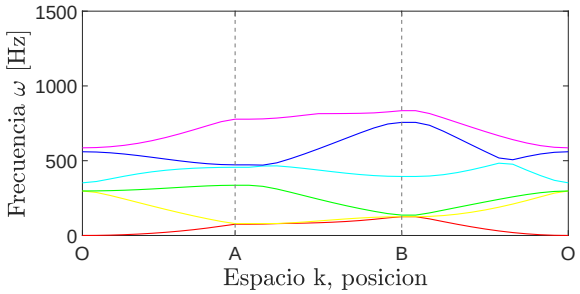
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.



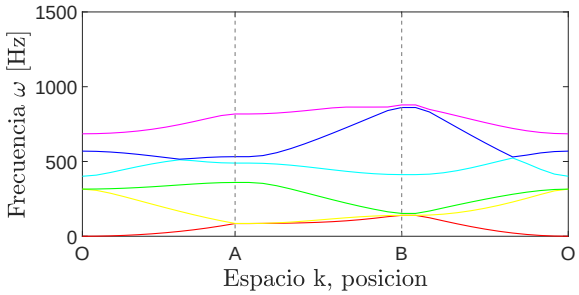
(c) Tercer ejemplo de diagrama de bandas.



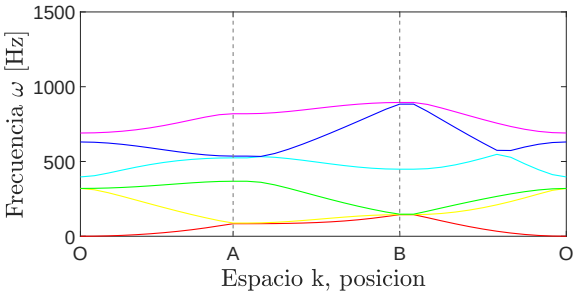
(d) Cuarto ejemplo de diagrama de bandas.

Figura D.10: Diagramas de bandas para placas modeladas con IGA y optimizadas con MBPSO para $n = 32$.

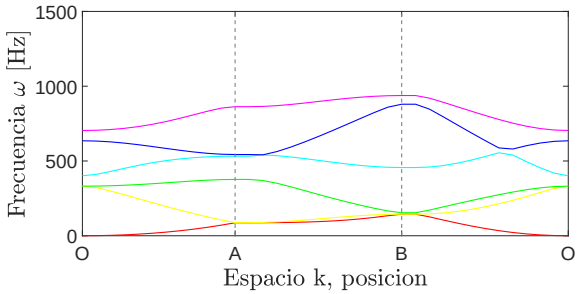
En las figuras D.11 y D.12 se muestran los diagramas de bandas para las placas resultantes modeladas con IGA, optimizadas con MpGA para $n = 16$ y $n = 32$ respectivamente.



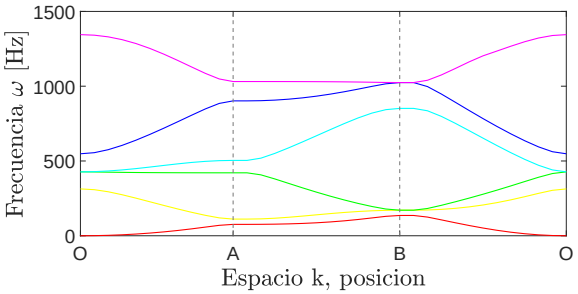
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.

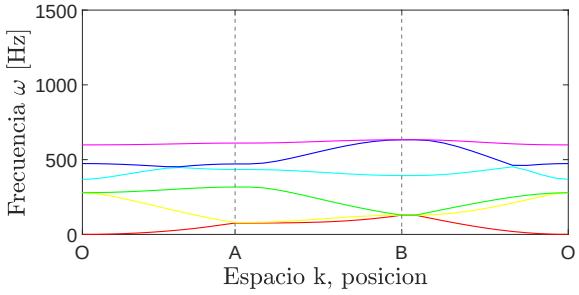


(c) Tercer ejemplo de diagrama de bandas.

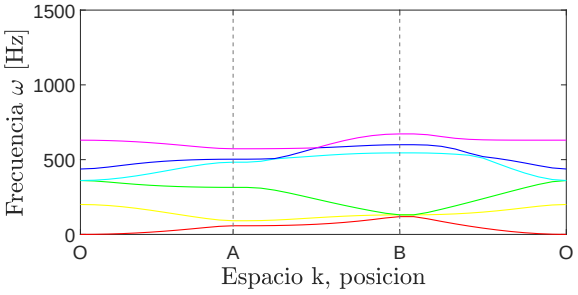


(d) Cuarto ejemplo de diagrama de bandas.

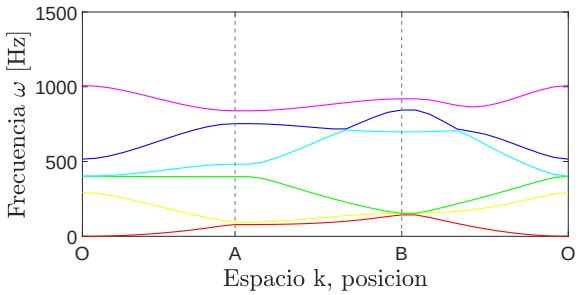
Figura D.11: Diagramas de bandas para placas modeladas con IGA y optimizadas con MBPSO para $n = 16$.



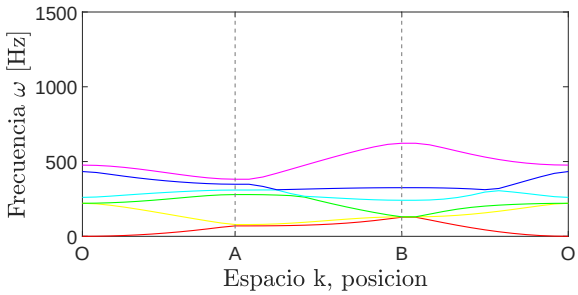
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.



(c) Tercer ejemplo de diagrama de bandas.

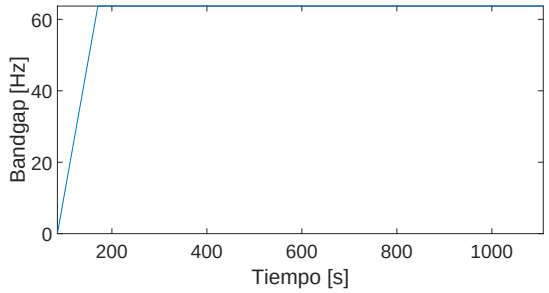


(d) Cuarto ejemplo de diagrama de bandas.

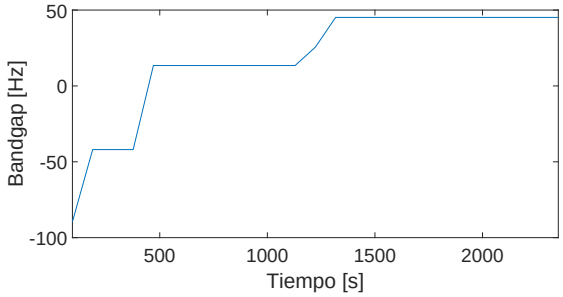
Figura D.12: Diagramas de bandas para placas modeladas con IGA y optimizadas con MBPSO para $n = 32$.

D.3. Curvas de Convergencia

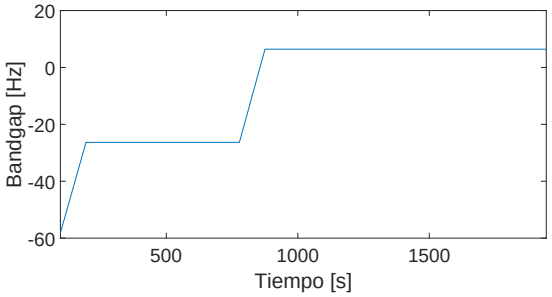
En las figuras D.13 y D.14 se presentan las curvas de convergencia resultantes de la optimización usando el modelo FEA y para placas de 16 y 32 elementos por arista respectivamente, optimizadas con el algoritmo MBPSO.



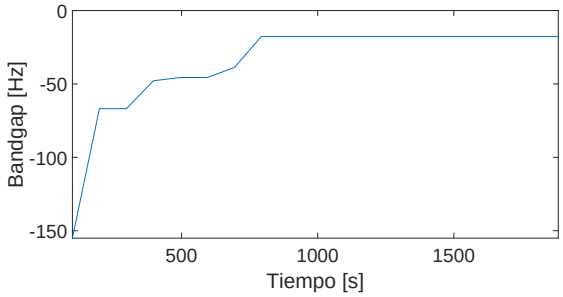
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.

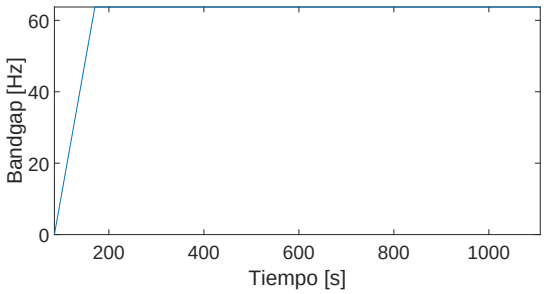


(c) Tercer ejemplo de diagrama de bandas.

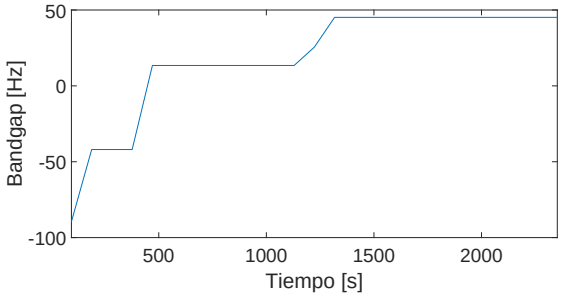


(d) Cuarto ejemplo de diagrama de bandas.

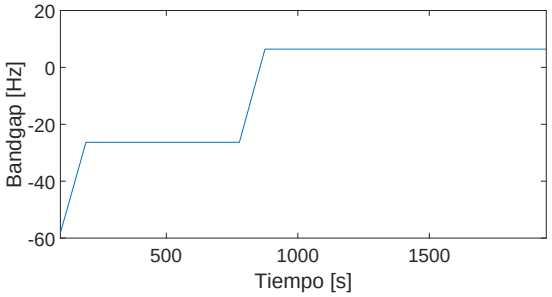
Figura D.13: Curvas de convergencia para placas modeladas con FEA y optimizadas con MBPSO para $n = 16$.



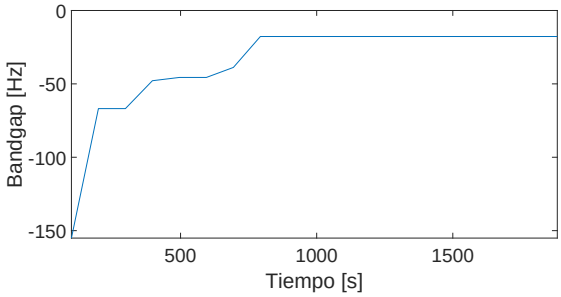
(a) Primer ejemplo de diagrama de bandas.



(b) Segundo ejemplo de diagrama de bandas.



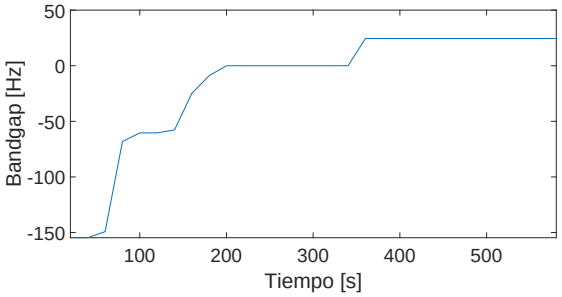
(c) Tercer ejemplo de diagrama de bandas.



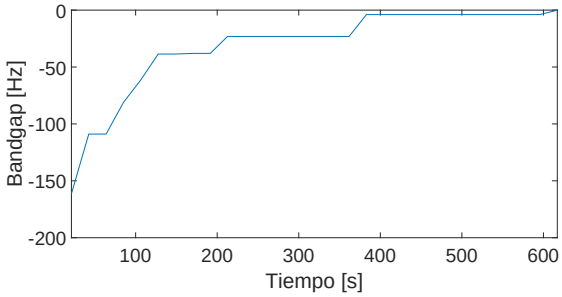
(d) Cuarto ejemplo de diagrama de bandas.

Figura D.14: Curvas de convergencia para placas modeladas con FEA y optimizadas con MBPSO para $n = 32$.

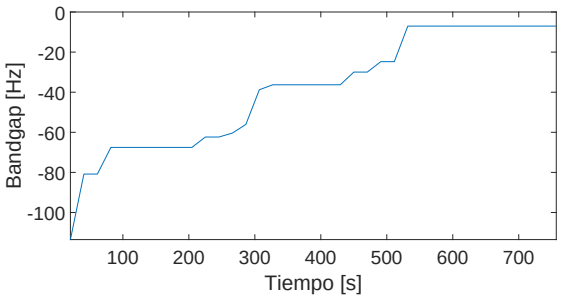
En las figuras D.15 y D.16 se presentan las curvas de convergencia resultantes de la optimización usando el modelo IGA y para placas de 16 y 32 elementos por arista respectivamente, optimizadas con el algoritmo MBPSO.



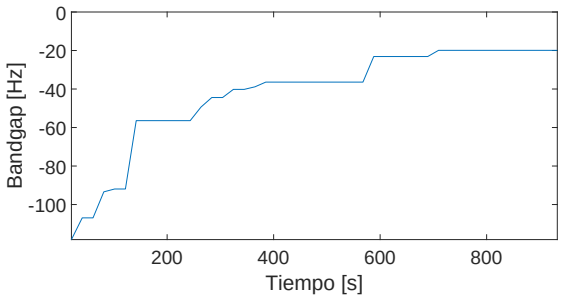
(a) Curva de convergencia para la primera repetición.



(b) Curva de convergencia para la segunda repetición.

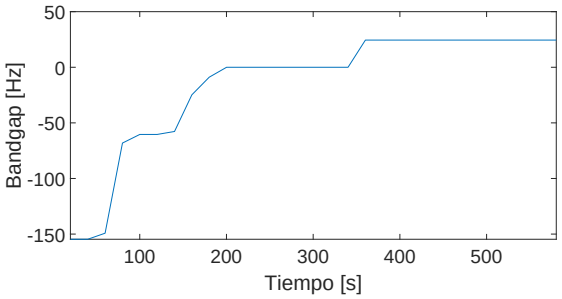


(c) Curva de convergencia para la tercera repetición.

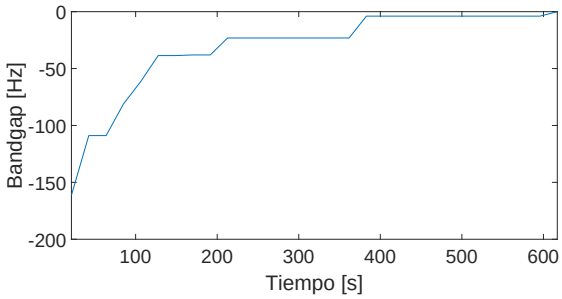


(d) Curva de convergencia para la cuarta repetición.

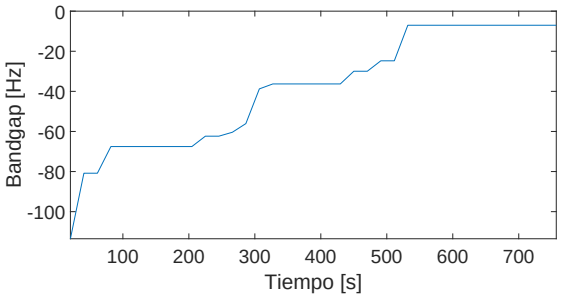
Figura D.15: Curvas de convergencia para placas modeladas con IGA y optimizadas con MBPSO para $n = 16$.



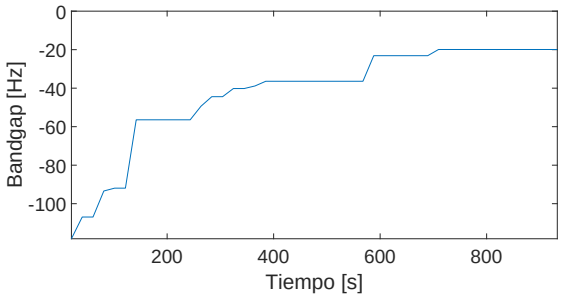
(a) Curva de convergencia para la primera repetición.



(b) Curva de convergencia para la segunda repetición.



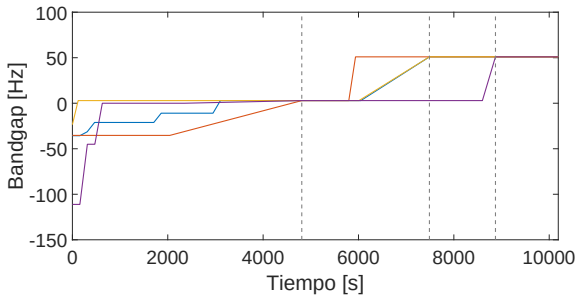
(c) Curva de convergencia para la tercera repetición.



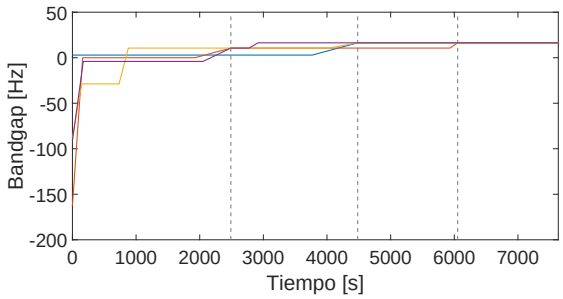
(d) Curva de convergencia para la cuarta repetición.

Figura D.16: Curvas de convergencia para placas modeladas con IGA y optimizadas con MBPSO para $n = 32$.

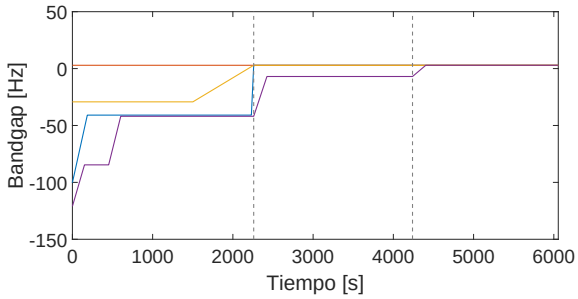
En las figuras D.17 y D.18 se presentan las curvas de convergencia resultantes de la optimización usando el modelo FEA y para placas de 16 y 32 elementos por arista respectivamente, optimizadas con el algoritmo MpGA.



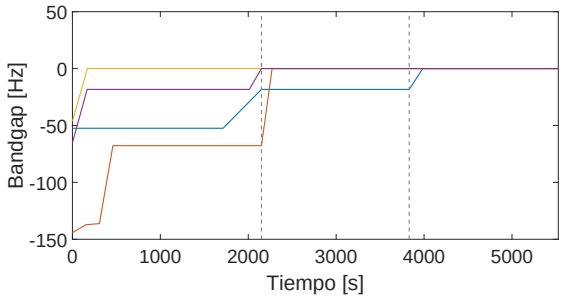
(a) Curva de convergencia para la primera repetición.



(b) Curva de convergencia para la segunda repetición.

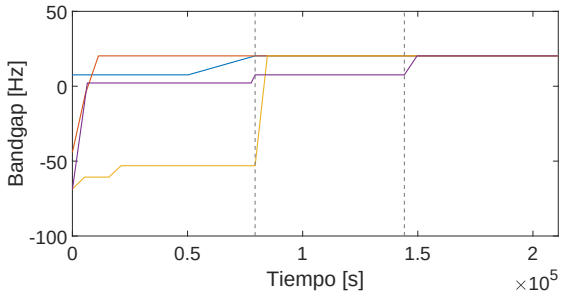


(c) Curva de convergencia para la tercera repetición.

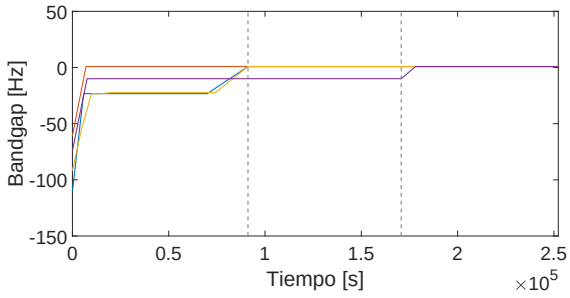


(d) Curva de convergencia para la cuarta repetición.

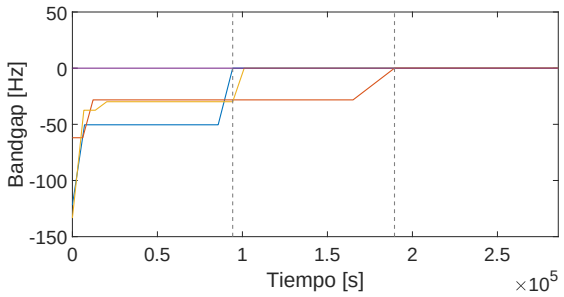
Figura D.17: Curvas de convergencia para placas modeladas con FEA y optimizadas con MpGA para $n = 16$.



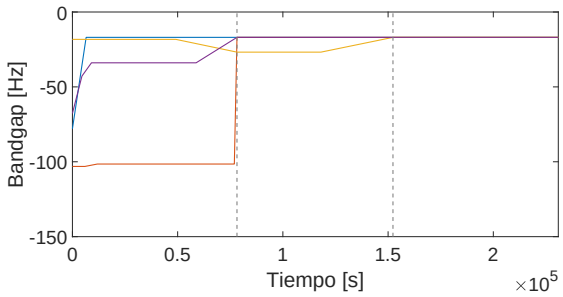
(a) Curva de convergencia para la primera repetición.



(b) Curva de convergencia para la segunda repetición.



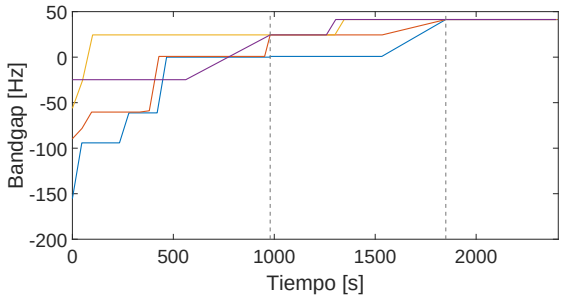
(c) Curva de convergencia para la tercera repetición.



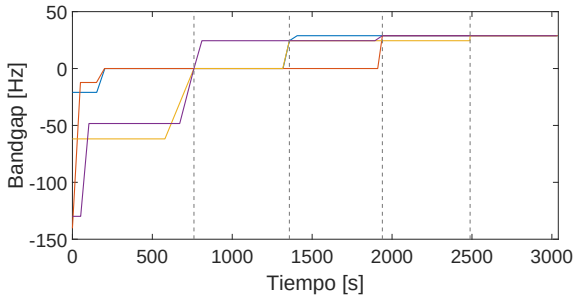
(d) Curva de convergencia para la cuarta repetición.

Figura D.18: Curvas de convergencia para placas modeladas con FEA y optimizadas con MpGA para $n = 32$.

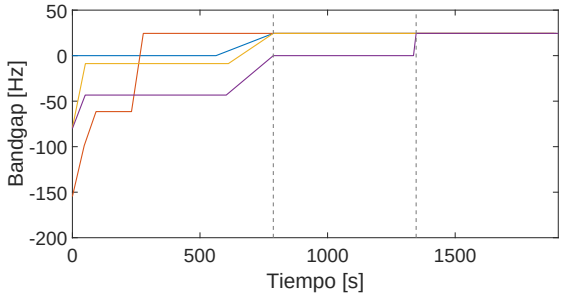
En las figuras D.19 y D.20 se presentan las curvas de convergencia resultantes de la optimización usando el modelo IGA y para placas de 16 y 32 elementos por arista respectivamente, optimizadas con el algoritmo MpGA.



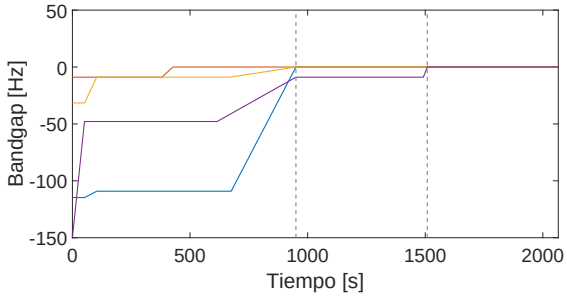
(a) Curva de convergencia para la primera repetición.



(b) Curva de convergencia para la segunda repetición.

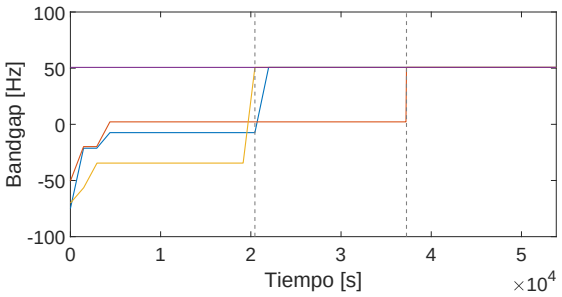


(c) Curva de convergencia para la tercera repetición.

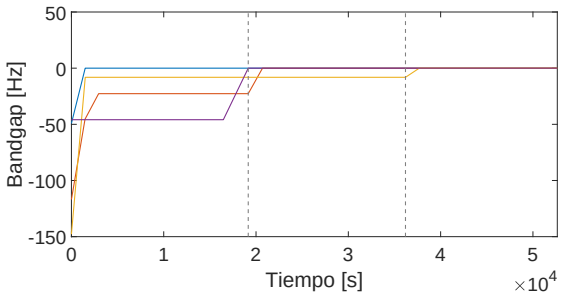


(d) Curva de convergencia para la cuarta repetición.

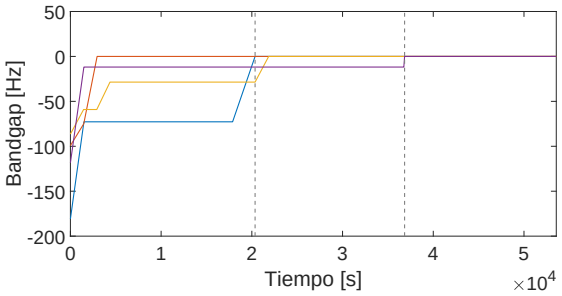
Figura D.19: Curvas de convergencia para placas modeladas con IGA y optimizadas con MpGA para $n = 16$.



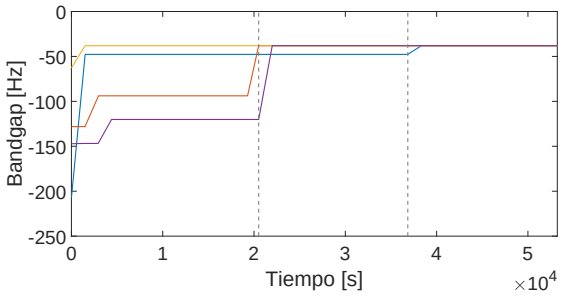
(a) Curva de convergencia para la primera repetición.



(b) Curva de convergencia para la segunda repetición.



(c) Curva de convergencia para la tercera repetición.



(d) Curva de convergencia para la cuarta repetición.

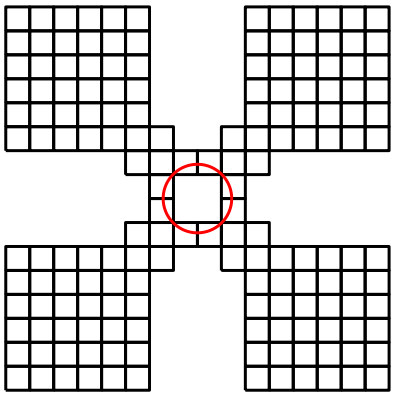
Figura D.20: Curvas de convergencia para placas modeladas con IGA y optimizadas con MpGA para $n = 16$.

Anexo E

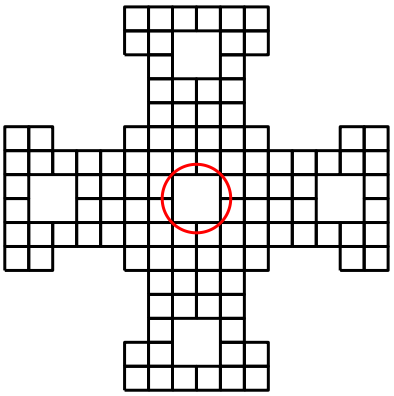
Resultados optimización en placas cuadradas con orificio circular

E.1. Placas resultantes

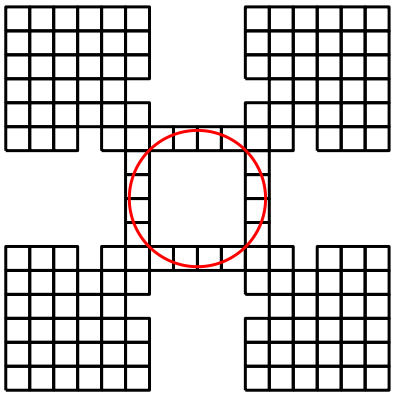
En las figuras E.1 a la E.2 se muestran algunos ejemplos de placas con orificio resultantes para placas con 16 y 32 elementos respectivamente.



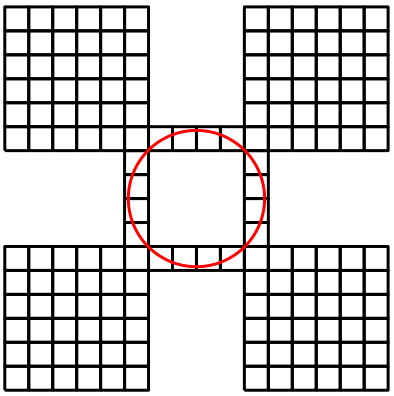
(a) Placa radio 0.0894m.



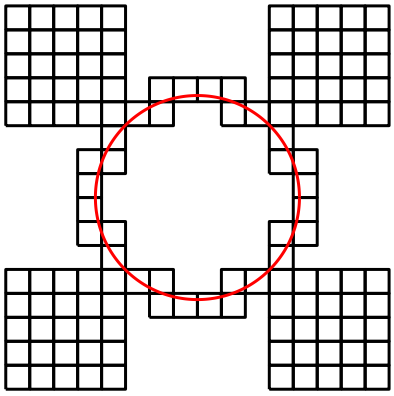
(b) Placa radio 0.0894m.



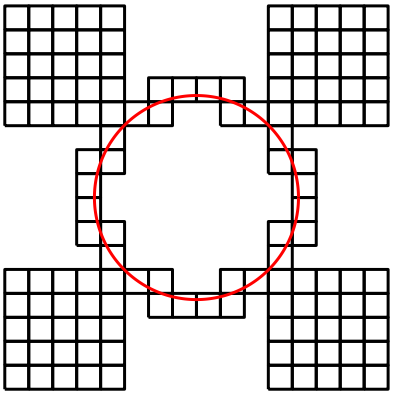
(c) Placa radio 0.1778m.



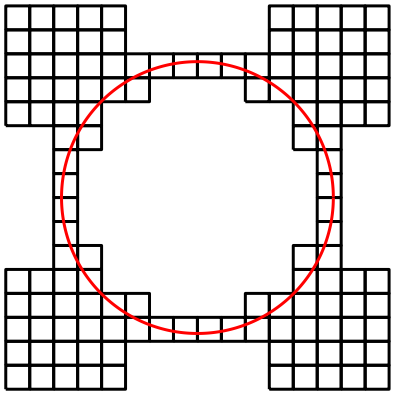
(d) Placa radio 0.1778m.



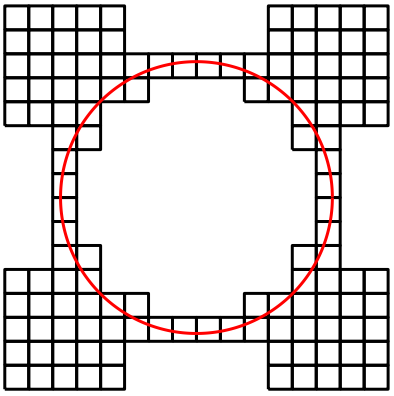
(e) Placa radio 0.2662m.



(f) Placa radio 0.2662m.

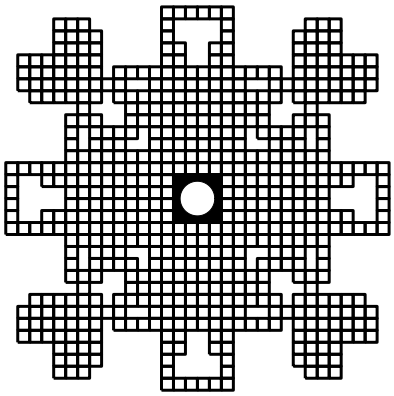


(g) Placa radio 0.3546m.

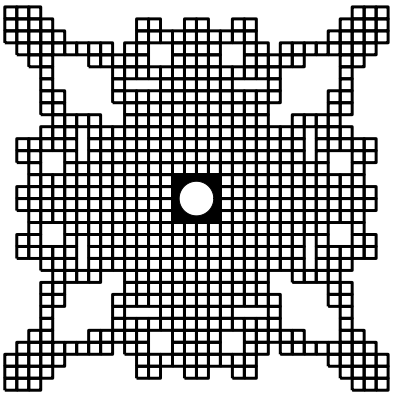


(h) Placa radio 0.3546m.

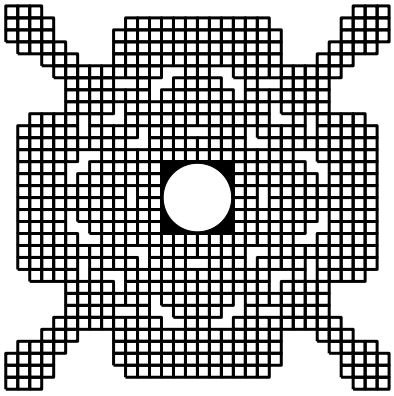
Figura E.1: Ejemplo de placas con orificio resultantes para $n = 16$ y diferentes radios.



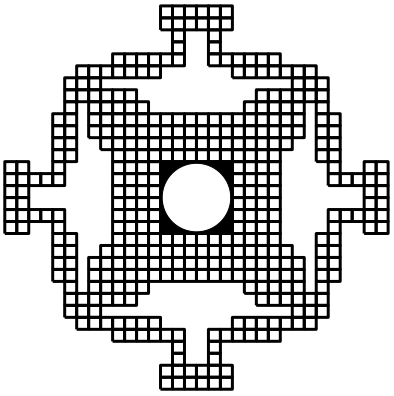
(a) Placa radio $0.0452m$.



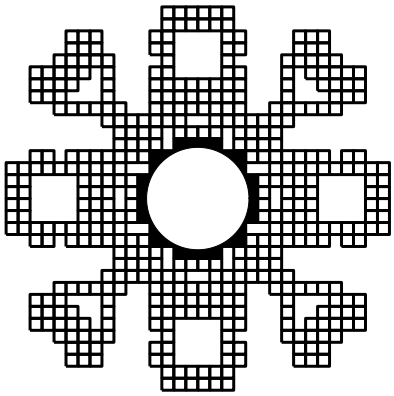
(b) Placa radio $0.0452m$.



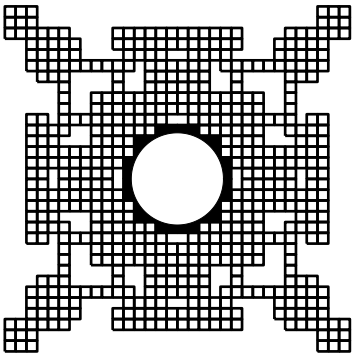
(c) Placa radio $0.0894m$.



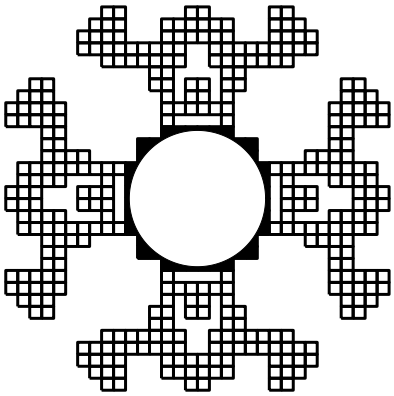
(d) Placa radio $0.0894m$.



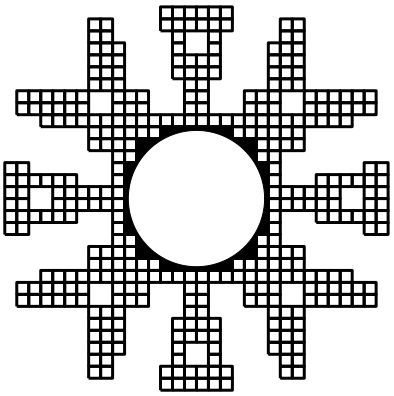
(e) Placa radio $0.1336m$.



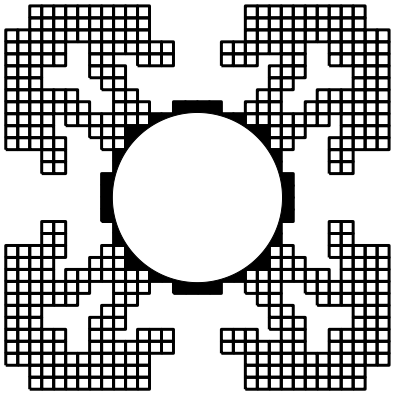
(f) Placa radio $0.1336m$.



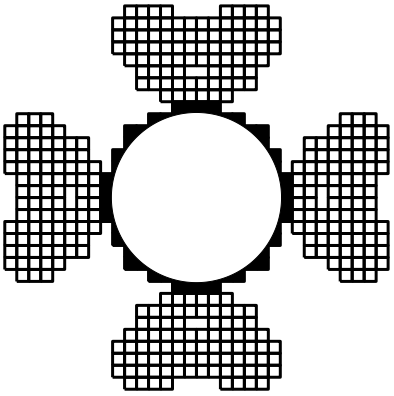
(g) Placa radio 0.1778m.



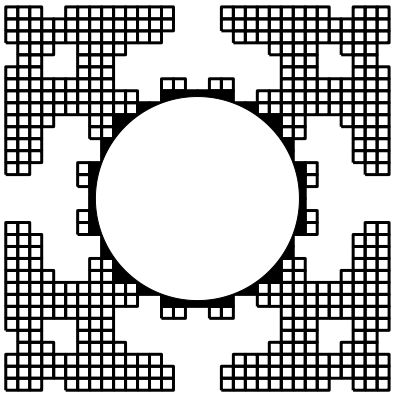
(h) Placa radio 0.1778m.



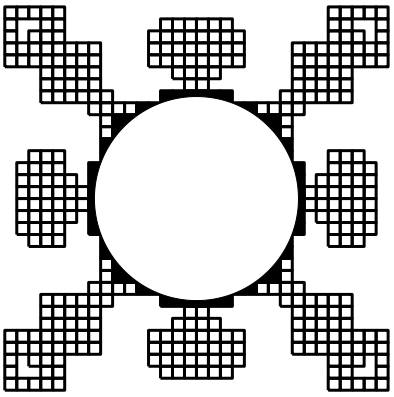
(i) Placa radio 0.2220m.



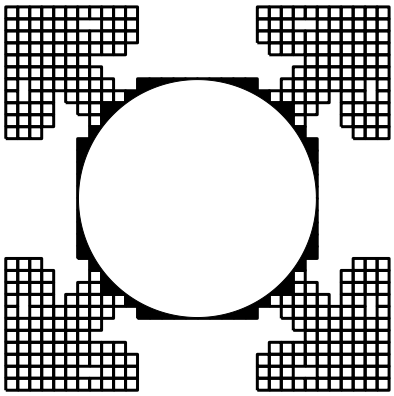
(j) Placa radio 0.2220m.



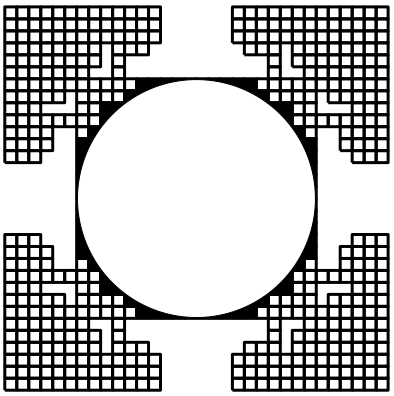
(k) Placa radio 0.2662m.



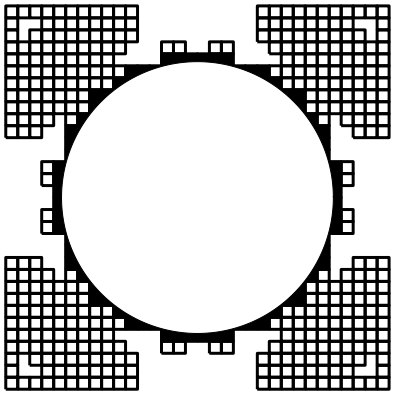
(l) Placa radio 0.2662m.



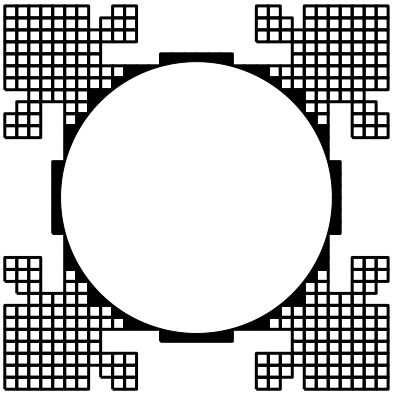
(m) Placa radio 0.3104m.



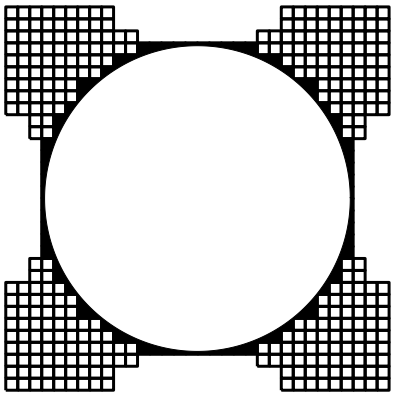
(n) Placa radio 0.3104m.



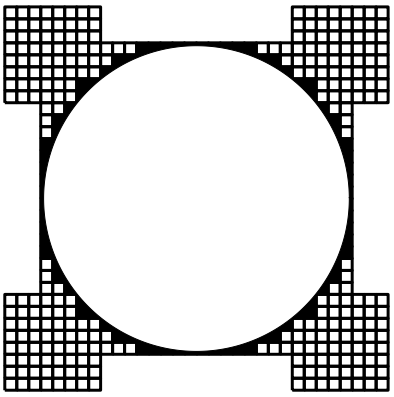
(ñ) Placa radio 0.3546m.



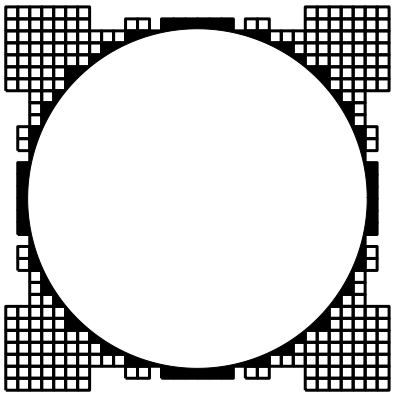
(o) Placa radio 0.3546m.



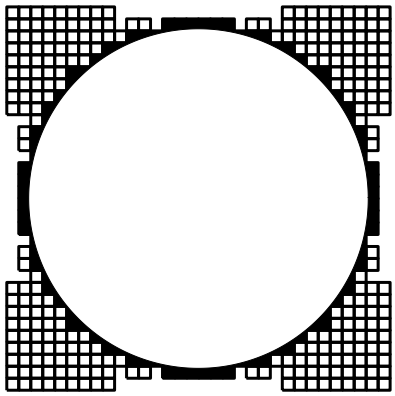
(p) Placa radio 0.3987m.



(q) Placa radio 0.3987m.



(r) Placa radio 0.4429m.

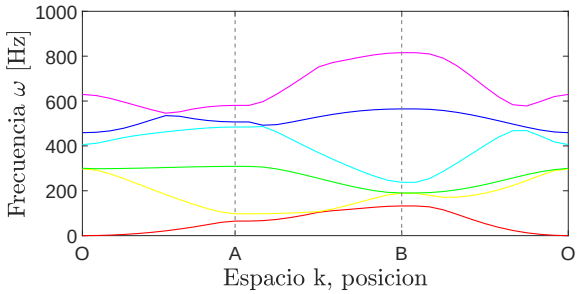


(s) Placa radio 0.4429m.

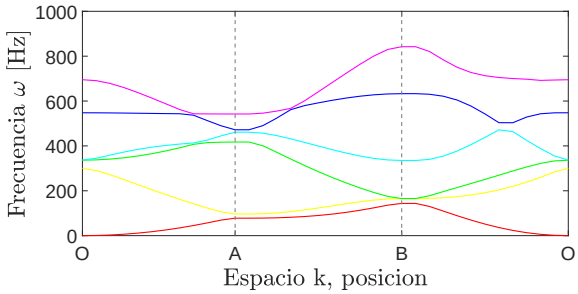
Figura E.2: Ejemplo de placas con orificio resultantes para $n = 32$ y diferentes radios.

E.2. Diagramas de Bandas

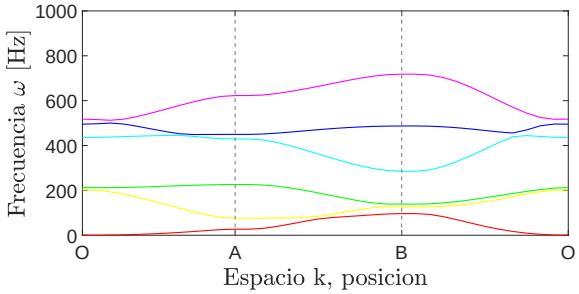
En las figuras E.3 a la E.4 se muestran algunos ejemplos de diagramas de bandas resultantes para placas con 16 y 32 elementos respectivamente.



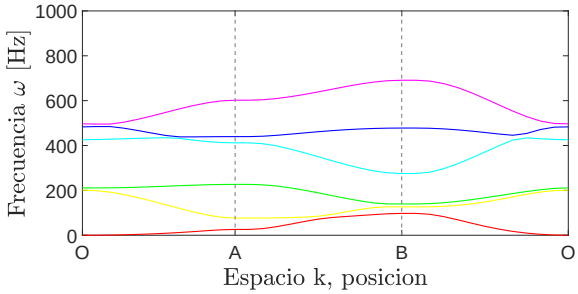
(a) Banda radio 0.0894m.



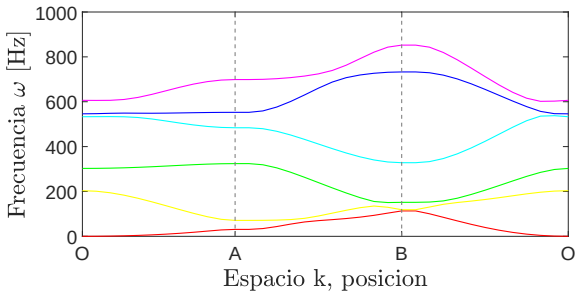
(b) Banda radio 0.0894m.



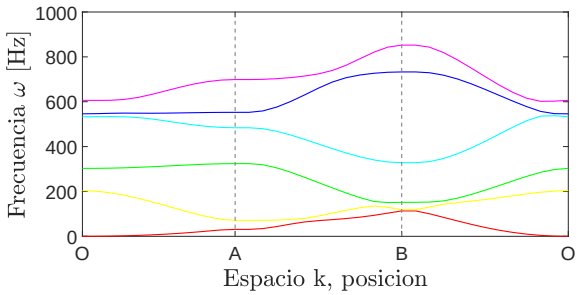
(c) Banda radio 0.1778m.



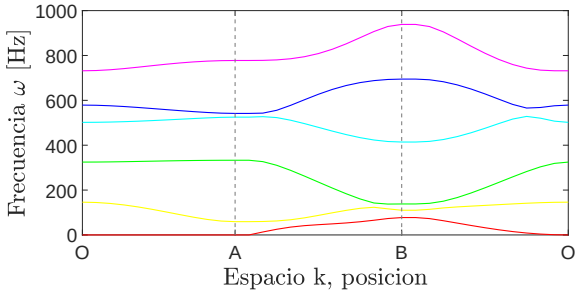
(d) Banda radio 0.1778m.



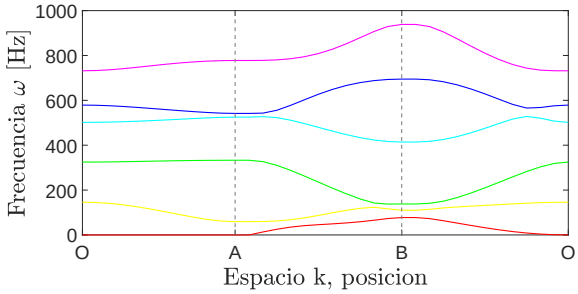
(e) Banda radio $0.2662m$.



(f) Banda radio $0.2662m$.

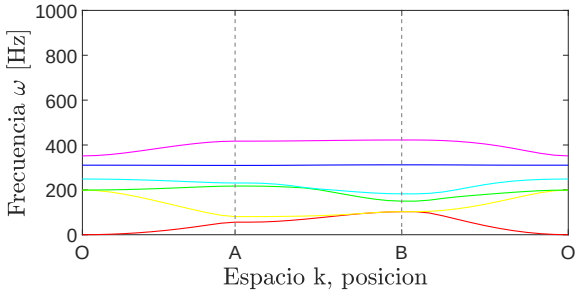


(g) Banda radio $0.3546m$.

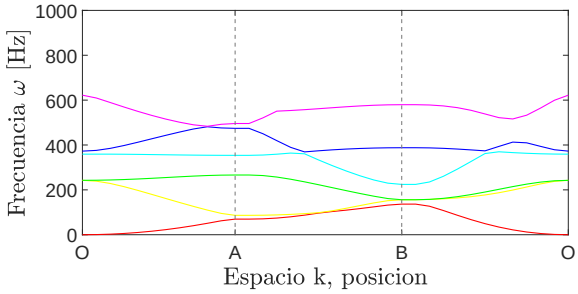


(h) Banda radio $0.3546m$.

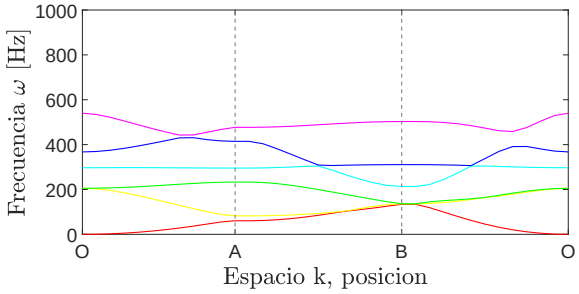
Figura E.3: Ejemplo de diagramas de bandas resultantes para $n = 16$ y diferentes radios.



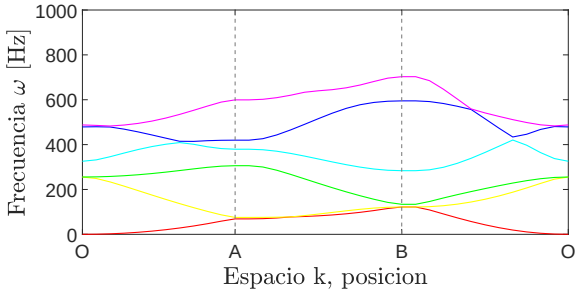
(a) Banda radio $0.0452m$.



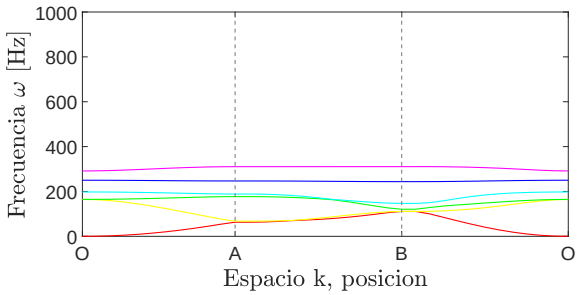
(b) Banda radio $0.0452m$.



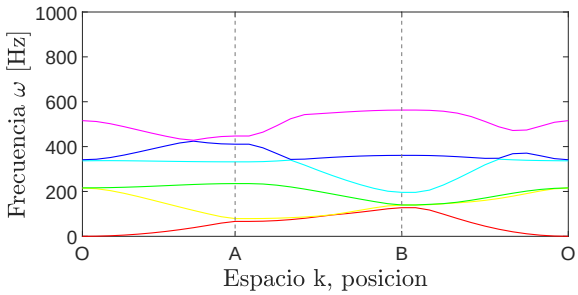
(c) Banda radio $0.0894m$.



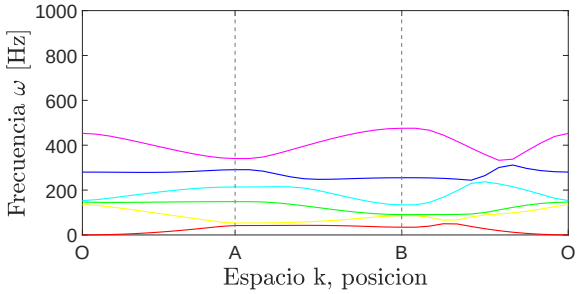
(d) Banda radio $0.0894m$.



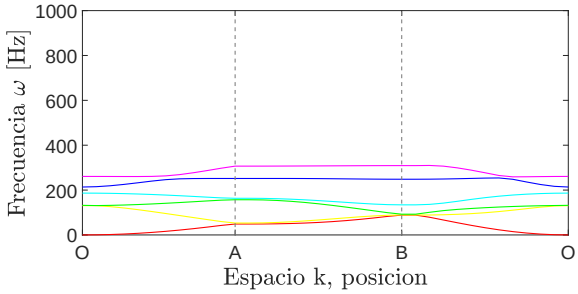
(e) Banda radio $0.1336m$.



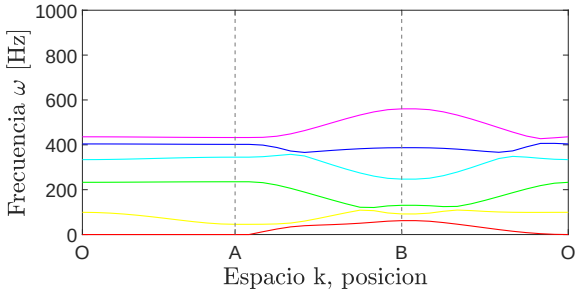
(f) Banda radio $0.1336m$.



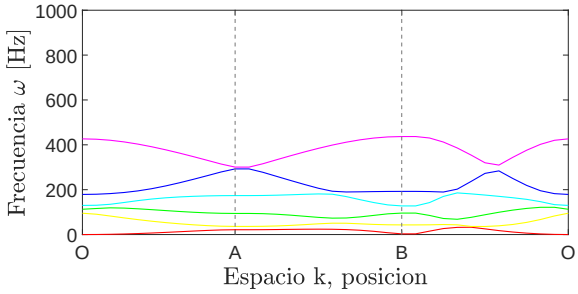
(g) Banda radio $0.1778m$.



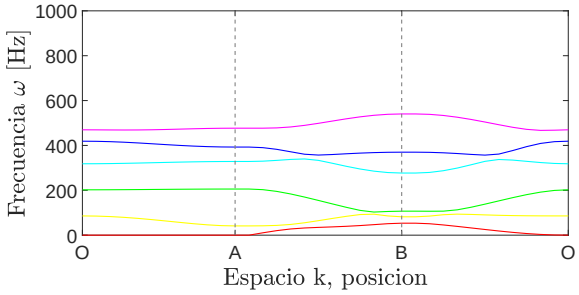
(h) Banda radio $0.1778m$.



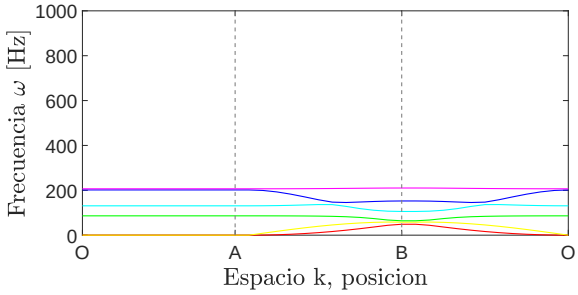
(i) Banda radio $0.2220m$.



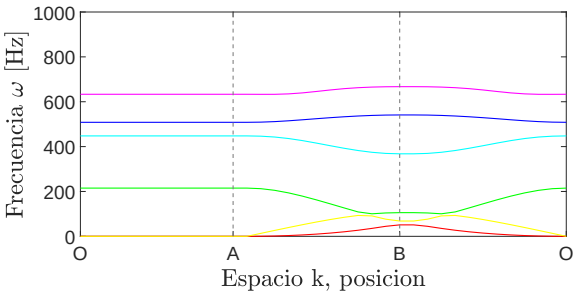
(j) Banda radio $0.2220m$.



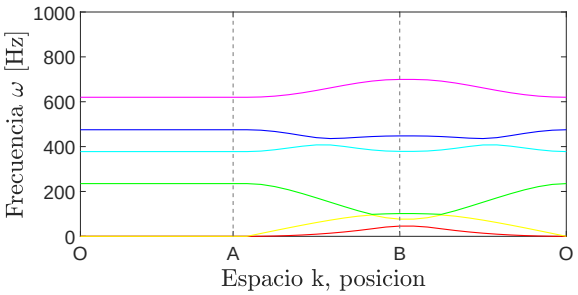
(k) Banda radio $0.2662m$.



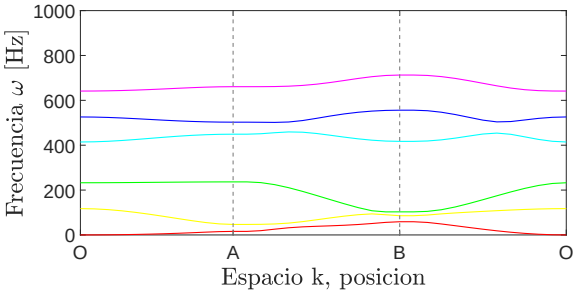
(l) Banda radio $0.2662m$.



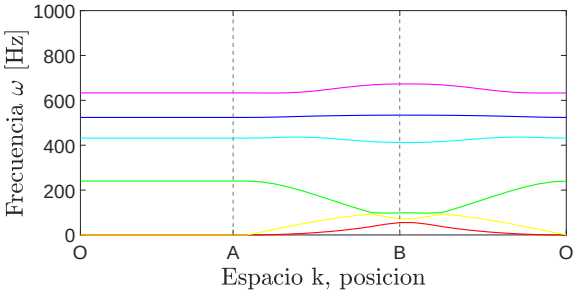
(m) Banda radio $0.3104m$.



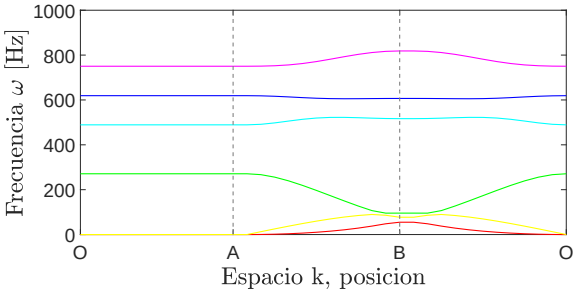
(n) Banda radio $0.3104m$.



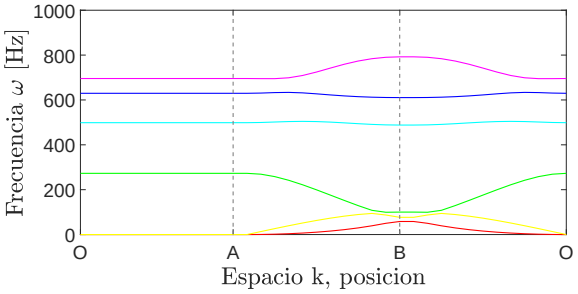
(ñ) Banda radio $0.3546m$.



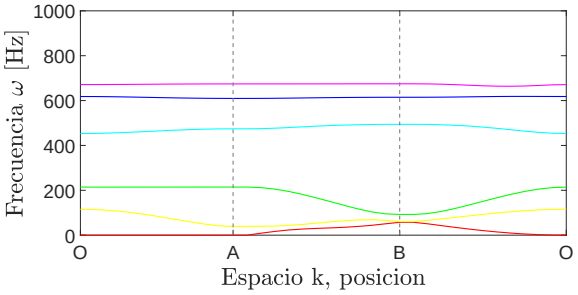
(o) Banda radio $0.3546m$.



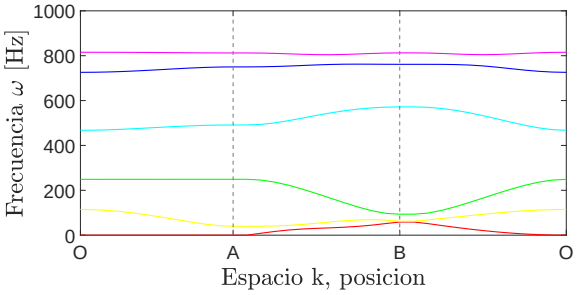
(p) Banda radio $0.3987m$.



(q) Banda radio $0.3987m$.



(r) Banda radio $0.4429m$.

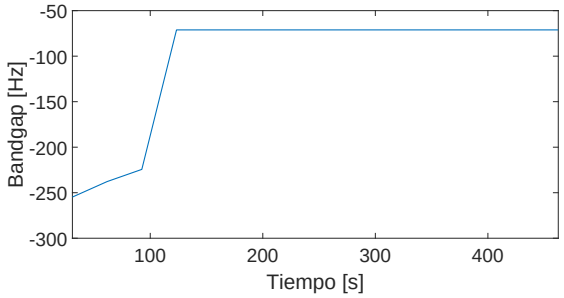


(s) Banda radio $0.4429m$.

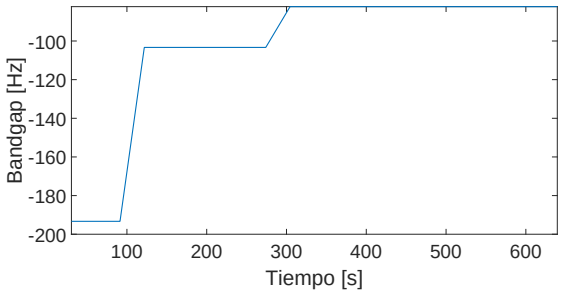
Figura E.4: Ejemplo de diagramas de bandas resultantes para $n = 32$ y diferentes radios.

E.3. Curvas de Convergencia

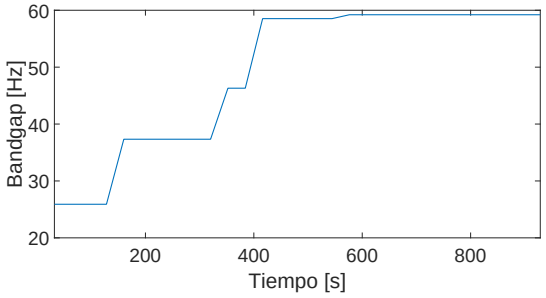
En las figuras E.5 a la E.6 se muestran algunos ejemplos de curvas de convergencia resultantes para placas con 16 y 32 elementos respectivamente.



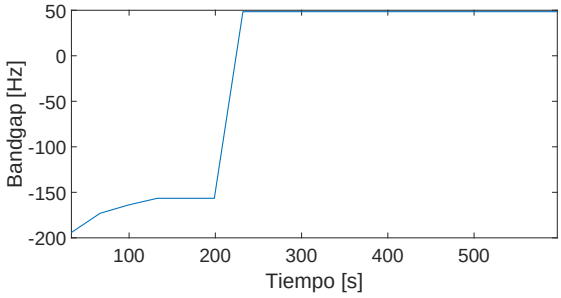
(a) Banda radio $0.0894m$.



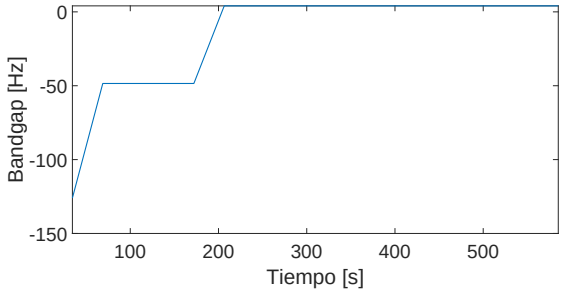
(b) Banda radio $0.0894m$.



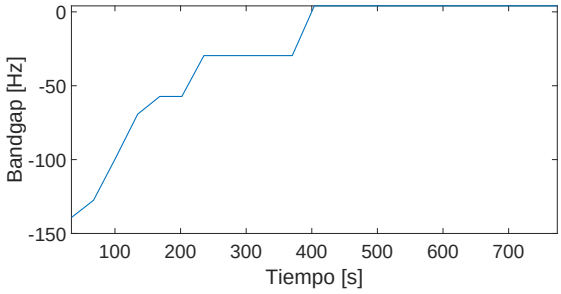
(c) Banda radio $0.1778m$.



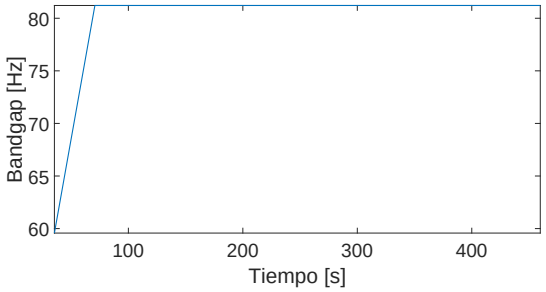
(d) Banda radio $0.1778m$.



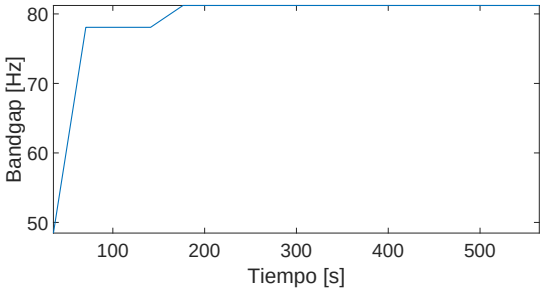
(e) Banda radio $0.2662m$.



(f) Banda radio $0.2662m$.

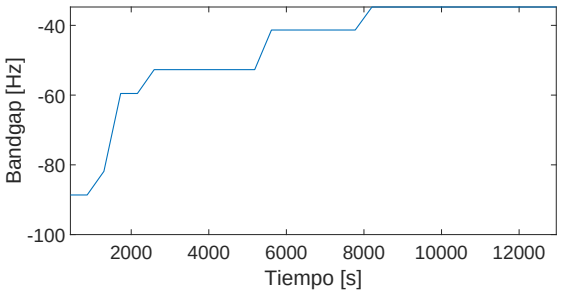


(g) Banda radio $0.3546m$.

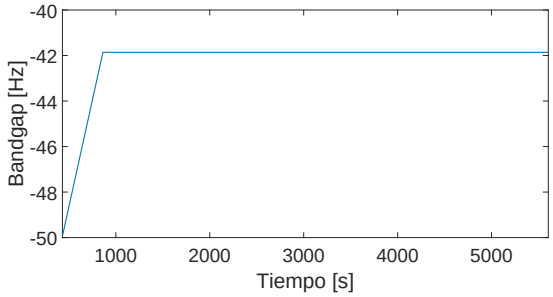


(h) Banda radio $0.3546m$.

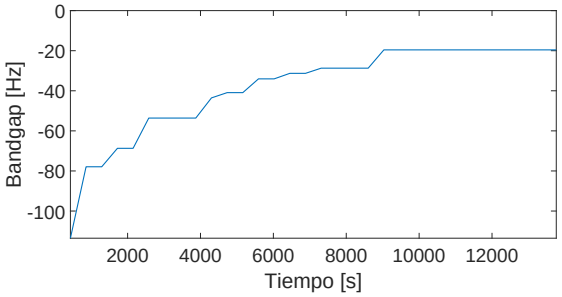
Figura E.5: Ejemplo de curvas de convergencia resultantes para $n = 16$ y diferentes radios.



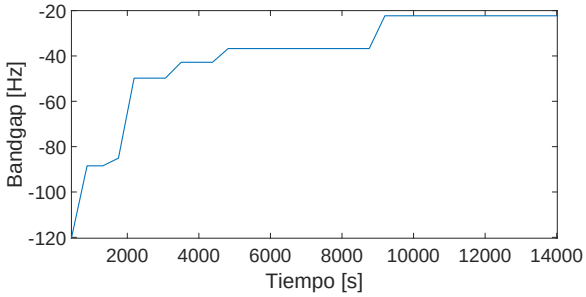
(a) Banda radio $0.0452m$.



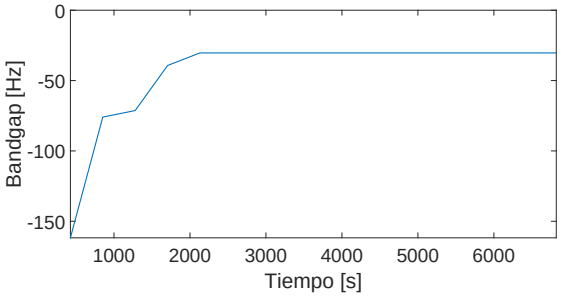
(b) Banda radio $0.0452m$.



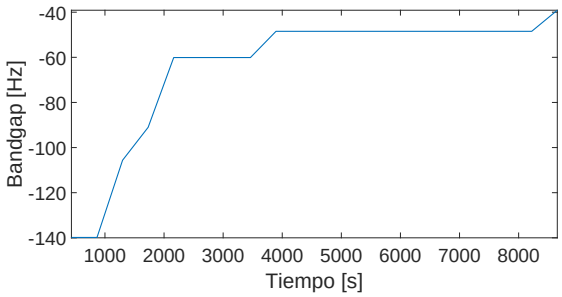
(c) Banda radio $0.0894m$.



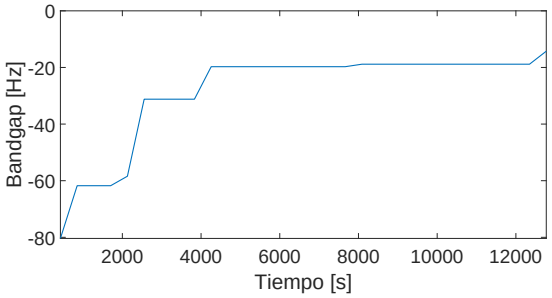
(d) Banda radio $0.0894m$.



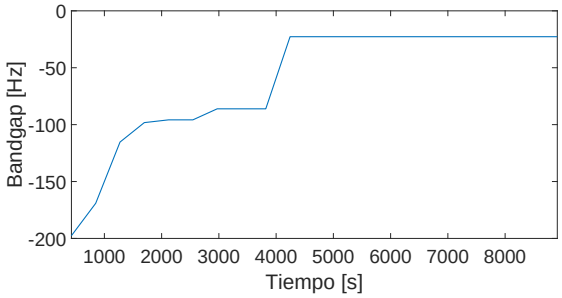
(e) Banda radio $0.1336m$.



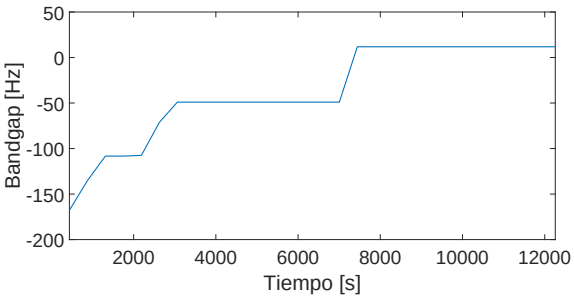
(f) Banda radio $0.1336m$.



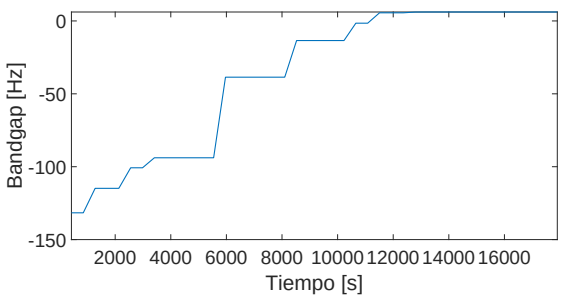
(g) Banda radio $0.1778m$.



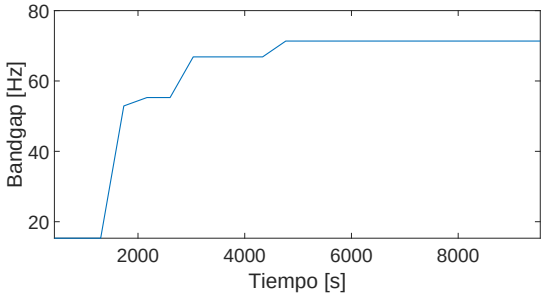
(h) Banda radio $0.1778m$.



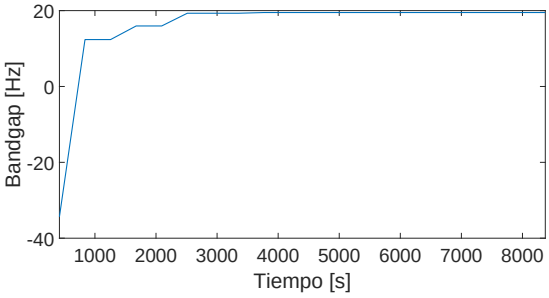
(i) Banda radio $0.2220m$.



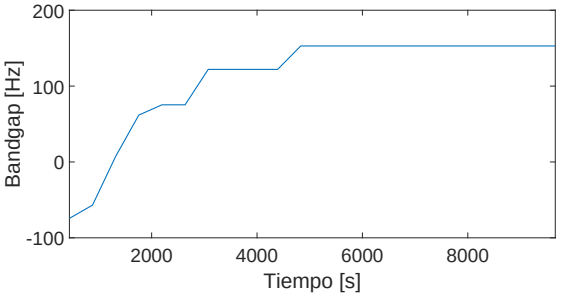
(j) Banda radio $0.2220m$.



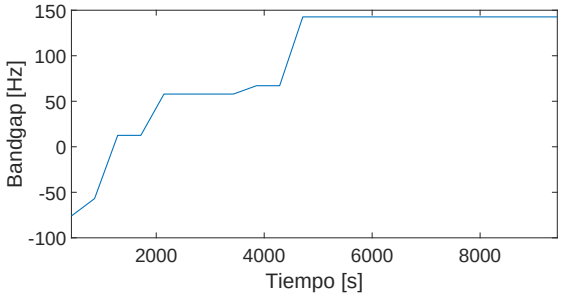
(k) Banda radio $0.2662m$.



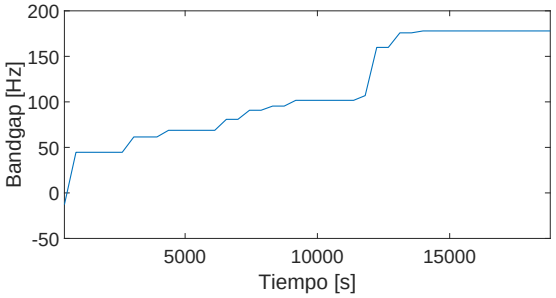
(l) Banda radio $0.2662m$.



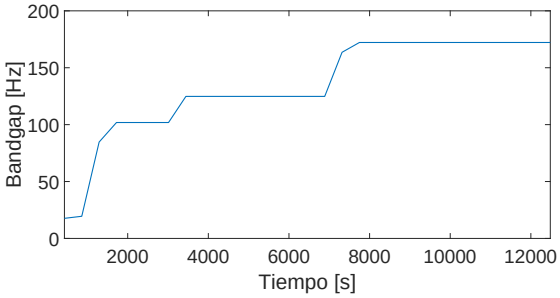
(m) Banda radio $0.31044m$.



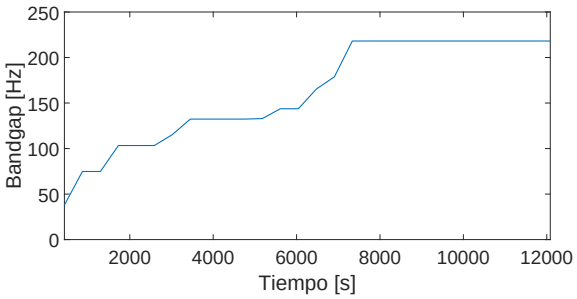
(n) Banda radio $0.3104m$.



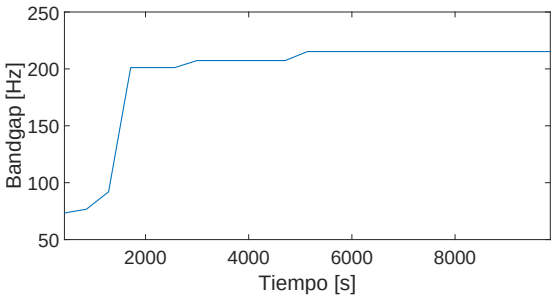
(ñ) Banda radio $0.3546m$.



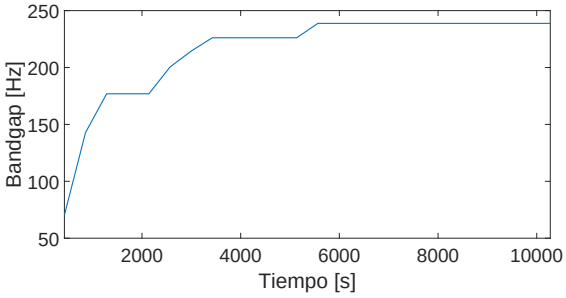
(o) Banda radio $0.3546m$.



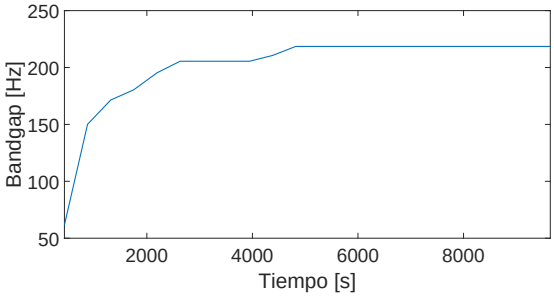
(p) Banda radio $0.3987m$.



(q) Banda radio $0.3987m$.



(r) Banda radio $0.4429m$.



(s) Banda radio $0.4429m$.

Figura E.6: Ejemplo de curvas de convergencia resultantes para $n = 32$ y diferentes radios.