UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

"Mat. Luis Manuel Rivera Gutiérrez"

ESTUDIO NUMÉRICO DE LA PROPAGACIÓN DE LA LUZ EN GUÍAS DE ONDAS DE CRISTAL FOTÓNICO UTILIZANDO EL MÉTODO DE LA ECUACIÓN INTEGRAL Y EL MÉTODO DE ELEMENTOS FINITOS

TESIS

Para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA FÍSICA

PRESENTA:

ELIEZER LOZANO TREJO

ASESOR DE TESIS:
Doctor en Ciencias en Óptica
DR. HÉCTOR I. PÉREZ AGUILAR

CO-ASESOR DE TESIS:
Doctor en Ciencias en Matemáticas
DR. FRANCISCO DOMÍNGUEZ MOTA

Morelia, Michoacán, Julio de 2019

RESUMEN

En los últimos 50 años el avance en la tecnología de los semiconductores ha sido tan grande que nos permitió pasar de procesadores con algunos miles de transistores en su interior a procesadores con miles de millones de transistores a precios accesibles para la mayoría de las personas. Esto ha impulsado el avance en otros campos de las ciencias como la medicina, la biología, la química, la física, etcétera. Todo esto ha ayudado a mejorar la calidad de vida de la humanidad. Sin embargo, la tecnología a base de silicio está alcanzando sus límites y para tratar de seguir impulsando a las demás ciencias los investigadores han centrado sus ojos en la nanofotónica, la cual tiene el potencial de reemplazar la tecnología a base de silicio. Ésta consiste en estudiar las interacciones de la materia con la luz en escala nanométrica, así como la fabricación de materiales nanoestructurados, los cuales nos permiten modificar las propiedades físicas de la luz. Uno de estos materiales nanoestructurados son los Cristales Fotónicos o más concretamente las guías de ondas de cristal fotónico (PCWs), las cuales son estudiadas en este proyecto de investigación.

En este trabajo nos centramos en estudiar la respuesta óptica de una PCW mediante las técnicas numéricas conocidas como el Método de la Ecuación Integral (IEM) (bajo la programación en paralelo con MPI, ScaLAPACK, OpenMP y CUDA) y el Método de Elementos Finitos (FEM). En la primera parte del trabajo se utilizaron las herramientas de cómputo en paralelo, previamente mencionadas, para mejorar el rendimiento del IEM al calcular la respuesta óptica de una PCW bidimensional de conductor real, usando tanto procesadores como la tarjeta gráfica y una combinación de ambos; así como en la forma secuencial para hacer un comparativo del tiempo de cómputo variando el número de periodos de la PCW. La mejor opción se obtuvo mediante la GPU con una rapidez de 1406 veces más rápido que con la versión secuencial. En la segunda parte se desarrolla el FEM en combinación con la técnica PML para resolver numéricamente la ecuación de Helmholtz para problemas con dominios abiertos. Aplicamos esta técnica a algunos problemas de ondas estacionarias en 1D y 2D para diferentes geometrías y así obtener las soluciones numéricas utilizando nuestros propios algoritmos. Finalmente, con ambos métodos numéricos presentamos la respuesta óptica mediante el campo cercano de la PCW considerada mostrando una buena correspondencia.

Palabras clave: IEM, FEM, PCW, MPI, ScaLAPACK, CUDA.

ABSTRACT

In the last 50 years the advance in semiconductor technology has been so great that it has allowed us to pass from processors with some of the transistors inside them to a processor of billions of transistors at affordable prices for most people. This has spurred progress in other fields of sciences such as medicine, biology, chemistry, physics and etc. All this has helped to improve the quality of life of humanity. However, silicon-based technology is reaching its limits. Researchers have focused their eyes on nanophotonics so they can keep impulsing the other sciences, which have the potential to replace silicon-based technology. This consists of studying the interactions of matter with light at a nanometric scale, as well as the manufacture of nanostructured materials, which allow us to modify the physical properties of light. One of these nanostructured materials are the Photonic Crystals or more specifically the photonic crystal waveguides (PCWs), which are studied in this research project.

In this investigation we focus on studying the optical response of a PCW using the numerical techniques known as the Integral Equation Method (IEM) (using parallel programming with MPI, ScaLAPACK, OpenMP and CUDA) and the Finite Element Method (FEM). In the first part of the work, the previously mentioned parallel computing tools were used to improve the performance of the IEM when calculating the optical response of the two-dimensional PCW, using processors as well as the graphics card and a combination of both; as well as in the sequential form to make a comparative of the computation time varying the number of periods of the PCW. The best option was obtained through the GPU with a speed of 1406 times faster than the sequential version. In the second part, the FEM is developed in combination with the PML technique to numerically solve the Helmholtz equation for problems with open domains. We apply this technique to some problems of standing waves in 1D and 2D for different geometries and thus obtaining the numerical solutions using our own algorithms. Finally, with both numerical methods, we present the optical response through the near field of the PCW considered showing a good correspondence.

Keywords: IEM, FEM, MPI, ScaLAPACK, CUDA.

Dedicatorias

Mi dedicación especial en el presente trabajo de tesis está dirigida a mis padres Noé Lozano Laurel y Salud Trejo Cabrera quienes me dieron la vida y han estado conmigo en todo momento. Gracias por todo papá y mamá por darme una carrera para mi futuro y por creer en mí.

También dedico este trabajo a mis tíos Wilbert Molina Toledo y Yazmin Trejo Cabrera, así como a mi hermana Itzanami quien me ayudo a terminar esta tesis y a todos mis familiares quienes me apoyaron todo el tiempo.

Así como a mis amistades más sinceras que estuvieron conmigo en los momentos buenos y malos.

A mis maestros quienes nunca desistieron al enseñarme, a los sinodales quienes estudiaron mi tesis y la aprobaron.

A todos los que me apoyaron para escribir y concluir esta tesis.

Sinceramente
ELIEZER LOZANO TREJO

Agradecimientos

En primer lugar deseo agradecer sinceramente a mi asesor de tesis, Dr. Héctor I. Pérez Aguilar, por su esfuerzo y dedicación.

Agradezco en forma especial al Dr. Francisco Domínguez Mota al aceptar participar co-asesorando mi trabajo de tesis.

Al Consejo Nacional de Ciencias y Tecnología (CONACYT) por el apoyo económico brindado durante mis estudios de posgrado.

Agradezco en forma especial a los sinodales, quienes me hicieron las correcciones y observaciones de mi trabajo de tesis.

A la División de Estudios de Posgrado de la Facultad de Ciencias Físico-Matemáticas y a sus profesores quienes desinteresadamente compartieron sus conocimientos y experiencias para beneficio de mi persona.

De igual forma deseo agradecer a mis padres y familiares por el apoyo tanto moral y económico para seguir estudiando y lograr el objetivo trazado.

También agradezco a todos mis amigos que ayudaron a terminar esta tesis.

Atentamente
ELIEZER LOZANO TREJO

Contenido

Página

Contenido (continuación)

Página

Contenido (continuación)

Página

Lista de Figuras

Lista de Figuras (continuación)

Capítulo I

INTRODUCCIÓN

En las últimas décadas se ha visto un gran avance en el desarrollo tecnológico, el cual nos ha llevado a pasar de computadoras que funcionaban a base de tubos de vacío, a la aparición de los primeros microprocesadores a base de silicio en los años 70. Estos primeros procesadores estaban construidos con unos pocos de miles de transistores. El progreso tecnológico ha sido tan espectacular que al día de hoy se cuenta con procesadores, los cuales tienen en su interior alrededor de 10 mil millones de transistores cuyas dimensiones son de aproximadamente de 7 nm. Sin embargo, las tecnologías a base de silicio tienen una serie de problemas. Uno de los principales son las interconexiones metálicas que transportan la información dentro de los chips, las cuales disminuyen su capacidad a medida que el sistema se hace más pequeño (Cadien *et al.*, 2005). Esta limitación se ha hecho más evidente en los últimos años. Para esto podemos poner como ejemplo a la ley de Moore, la cual expresa que cada dos años se duplica el número de transistores en un microprocesador. Se trata de una ley empírica, cuyo cumplimiento se ha podido constatar hasta hoy (ver Fig. 1), pero se espera que ésta deje de ser válida para el año 2025. Para tratar de solucionar los problemas antes mencionados los investigadores han centrado su atención en la nanofotónica.

La propagación de la luz a través de un medio es sensible a la estructura del mismo, si

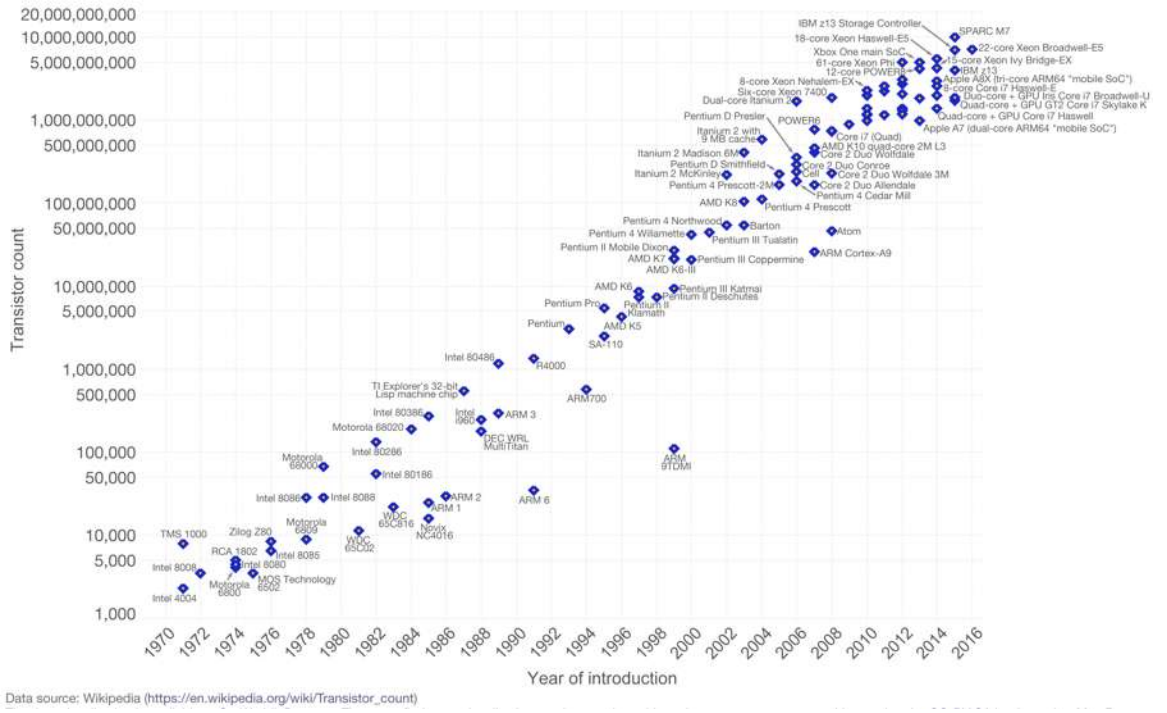

Figura 1: Ley de Moore. Número de transistores en un chip de circuito integrado (1971-2016).

la escala es del orden de la longitud de onda electromagnética incidente. De este modo, un mismo material puede interaccionar de forma diferente con la luz si se encuentra en forma de medio homogéneo, dispuesto periódicamente o de forma desordenada. Por ello, materiales dieléctricos con una estructura del mismo orden de la longitud de onda con la que se desea interactuar, se presentan como excelentes candidatos para fabricar componentes ópticos capaces de reflejar, confinar o guiar la luz del mismo modo que un metal, pero sin pérdidas debido a la absorción. A este tipo de materiales característicos se les conoce como los Cristales Fotónicos (PCs[1]).

Los Cristales Fotónicos son estructuras periódicas caracterizadas por la variación periódica del índice de refracción y la distribución espacial de la constante dieléctrica del material que las constituye (Joannopoulos *et al.*, 2008). Guardan numerosas similitudes

[1]Por sus siglas en inglés, Photonic Crystals.

con los cristales electrónicos en semiconductores, como el hecho de que ambos están formados a partir de la repetición de un elemento en el espacio, dando lugar a una red cristalina (McKelvey, 1966). Cuando la luz que se propaga en un PC interactúa con la modulación periódica del índice de refracción provoca una de las propiedades más relevantes de los PCs, las Bandas Prohibidas Fotónicas o gaps (PBGs[2]) (Sibilia *et al.*, 2009). Los PBGs son rangos de frecuencias para los cuales la propagación de ondas electromagnéticas no están permitidas en el interior del cristal. Por otro lado, las frecuencias en la cual es posible la propagación de la onda dentro del cristal se les denomina modos. Así, cuando la luz con una frecuencia perteneciente al PBG intente atravesar la estructura cristalina se verá casi totalmente reflejada; es decir, el cristal actuará como un espejo.

En general, los PCs se construyen con materiales semiconductores o dieléctricos, aunque en los últimos años los PCs metálicos o metal-dieléctrico han demostrado excepcionales propiedades ópticas. Estos sistemas se puede presentar en una o varias dimensiones del espacio, constituyendo así PCs en una (1DPC), dos (2DPC) o tres dimensiones (3DPC), como se aprecia en la Fig. 2.

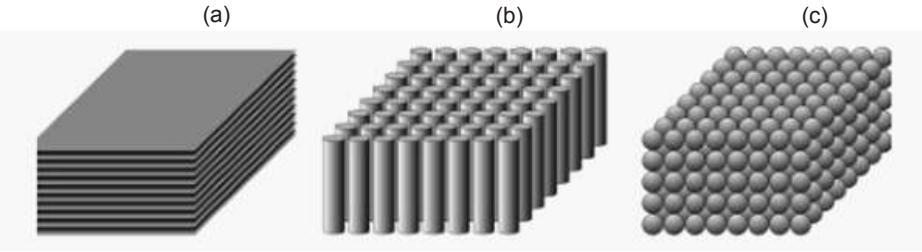

Figura 2: Ejemplos de una estructura fotónica unidimensional (a), bidimensional (b) y tridimensional (c).

Los PCs fueron propuestos a finales de los años ochenta cuando aparecieron en un par de artículos de Yablonovitch y John (Yablonovitch, 1987; John, 1987). Estos

[2]Por sus siglas en inglés, Photonic Band Gaps.

artículos tienen enfoques distintos: para Yablonovitch, los PCs eran un medio que permitía modificar la densidad de estados e inhibir la emisión espontánea; para John, los PCs permitirían la localización de luz. En 1991 se presenta el primer material con un gap fotónico a partir de una variación de la estructura diamante y que hoy se conoce como Yablanovita. Debido a que la periodicidad del sistema es de alrededor de la decena de mm, éste presenta su gap fotónico en el rango de las microondas. En 1996 Thomas Krauss (Krauss *et al.*, 1996), demostró que era posible obtener una estructura bidimensional en el rango cercano a frecuencias ópticas. De igual manera las investigaciones han estado encaminadas en lograr obtener un PC en el rango visible y en la fabricación en el rango de las microondas y en el infrarrojo cercano. Los 2DPCs encuentran su uso comercial en forma de Fibras de Cristal Fotónico (PCFs[3]) (también conocidas como fibras microestrucuradas). Cabe mencionar que en la naturaleza podemos encontrar cristales fotónicos naturales, como es el caso de los escarabajos, en las alas de las mariposas, en las plumas de los pavos reales o en los ópalos (ver Fig. 3).

Figura 3: Las alas de la mariposa, los escarabajos, las plumas de los pavo reales y los ópalos contienen microestructuras periódicas naturales que son responsables del color iridiscente. Dichas estructuras representan un cristal fotónico natural (Centeno Jiménez, 2014).

Una tecnología basada en el uso de la luz como portador de información tendría

[3]Por sus siglas en inglés, Photonic Crystal Fibre.

ventajas sobre la tecnología electrónica ya que los fotones, debido entre otros factores a la ausencia de carga y masa, resultan más eficientes que los electrones en el transporte de la información.

Para describir un PC es necesario conocer una serie de parámetros que determinarán sus propiedades fotónicas como son: parámetro de red, estructura cristalina, constante de red, topología, contraste de índice, periodo y fracción de llenado.

Parámetro de red. Debe ser del orden de la longitud de onda y determina el rango de frecuencias en el que aparecen las propiedades fotónicas.

Estructura cristalina. Está relacionada con la forma en que queda modulado el índice de refracción. Se caracteriza por tener un orden o periodicidad y puede adoptar estructuras en una, dos o las tres dimensiones del espacio. Podemos tener, por ejemplo, estructuras triangulares, cuadradas, etc (en 2D) o con simetría FCC[4], BCC[5] o diamante (en 3D).

Constante de red. Representa la longitud de los lados de la celda unitaria de una estructura cristalina. Pueden existir una, dos o hasta tres constantes de red distintas dependiendo del tipo de estructura de la que se trate. Este parámetro permite definir la relación entre las propiedades geométricas de una estructura con una de las propiedades físicas que caracterizan al PC que es su banda prohibida.

Topología. Una vez diseñada la estructura es muy importante como se realice la modulación periódica teniendo en cuenta la disposición de los centros esparcidores (zonas de alto índice de refracción). Así, podemos tener los centros esparcidores aislados entre ellos rodeados del medio de bajo índice, en lo que se denomina topología tipo Cermet. Al contrario, invirtiendo el papel de los centros esparcidores, estos pueden estar unidos entre sí formando una red, en lo que se conoce como topología tipo Network.

[4]Por sus siglas en inglés, Face Centered Cubic.
[5]Por sus siglas en inglés, Body Centered Cubic.

Como es de suponer, la distribución de campo en una y otra forma será muy diferente condicionando así las propiedades fotónicas de la estructura.

Contraste de índices. Es la razón entre el índice de refracción mayor y el menor. Cuanto mayor sea dicho contraste más marcadas serán las propiedades fotónicas.

Período. Es la distancia que separa una capa de un índice de refracción con la siguiente capa del mismo material, y está directamente relacionada con la longitud de onda de los fotones que se propagan.

Fracción de llenado. Es la razón entre el volumen ocupado por uno de los materiales con respecto al volumen total.

La combinación de todos estos factores hace que existan multitud de posibles estructuras con muy diversas propiedades fotónicas. Actualmente existen varios métodos numéricos que nos permiten estudiar los PCs; entre los que se destacan el método de ondas planas (PWE[6]) (Sözüer *et al.*, 1992), el método de Diferencias Finitas en Dominio del Tiempo (FDTD[7]) (Inan and Marshall, 2011), el método de Elementos Finitos (FEM[8]) (Mu *et al.*, 2015) y el método de la Ecuación Integral (IEM[9]) (Mendoza-Suárez *et al.*, 2006).

El IEM consiste en partir de las condiciones de frontera y del segundo teorema integral de Green, obteniéndose un par de ecuaciones integrales con las cuales se puede determinar el campo dentro y fuera de la guía de ondas en términos de un par de funciones fuente. Debido a que no es posible resolver de manera analítica el sistema de ecuaciones integrales acopladas, recurrimos a una discretización del sistema. Para esto es necesario convertir las ecuaciones integrales obtenidas en un sistema de ecuaciones algebraico que se puede resolver por medio de métodos numéricos matriciales. El número de ecuaciones algebraicas que se obtienen depende del número de puntos de muestreo que se requieren para construir los contornos de las regiones que se tienen en

[6]Por sus siglas en inglés, Plane Wave Expansion.
[7]Por sus siglas en inglés, Finite Diference Time Domain.
[8]Por sus siglas en inglés, Finite Element Method.
[9]Por sus siglas en inglés, Integral Equation Method.

el sistema.

El FEM está basado en la reformulación del problema en forma variacional y la solución del espacio de soluciones por otro de dimensión finita. Esto se logra al realizar una partición del dominio en elementos de tamaño finito y se busca soluciones mediante familias de funciones dependientes de un número finito de parámetros. De esta manera pasamos de un problema inicial a un sistema de ecuaciones que se puede resolver numéricamente.

La paralelización ha sido clave no sólo para aumentar el rendimiento en los problemas sino para reducir costos. Todo esto fue gracias a la creación de diferentes herramientas para el desarrollo de programas en paralelo que han ayudado a que esta actividad se vuelva sencilla. Algunas de estas herramientas son: MPI (Message Passing Interface) (Pacheco, 2011), OpenMP (Open Multi-Processing) (Chapman *et al.*, 2008), OpenCL (Open Computing Language) (Kaeli *et al.*, 2015) y CUDA (Computer Unified Device Architecture) (Cook, 2013).

La paralelización consiste en la subdivisión de un problema, en subproblemas, cada uno de los cuales es resuelto de forma simultánea y por separado, ofreciendo reducciones importantes en cuanto al tiempo de ejecución. Sin embargo, no todos los problemas son aptos para el empleo de la paralelización, y los que sí lo son, no tienen porqué responder de forma positiva a su resolución paralela.

En este trabajo nos centramos en estudiar la respuesta óptica de una PCW mediante las técnicas numéricas conocidas como el IEM y el FEM. En la primera parte del trabajo se utilizaron múltiples herramientas del cómputo en paralelo como son: OpenMP, MPI, ScaLAPACK y CUDA, para mejorar el rendimiento del IEM (Mendoza-Suárez and Pérez-Aguilar, 2016) al calcular la respuesta óptica de una PCW[10] bidimensional finita de conductor real, usando tanto procesadores como la tarjeta gráfica y una combinación

[10]Por sus siglas en inglés, Photonic Crystal Waveguides.

de ambos; así como en la forma secuencial para hacer un comparativo del tiempo de cómputo variando el número de periodos de la PCW. En la segunda parte se desarrolla el FEM en su formulación débil de Galerkin (Mu *et al.*, 2015) en combinación con la técnica PML[11] (Berenger, 1994; Bermudez *et al.*, 2006) para resolver numéricamente la ecuación de Helmholtz para problemas con dominios abiertos. Aplicamos esta técnica a algunos problemas de ondas estacionarias en 1D y 2D para diferentes geometrías y así obtener las soluciones numéricas. Finalmente, con ambos métodos numéricos presentamos la respuesta óptica mediante el campo cercano de la PCW.

I.1. Estructura de la tesis

Este trabajo de tesis está estructurado en ocho capítulos que a continuación describiremos.

En el capítulo II se presentarán los conceptos básicos relacionados con la programación en paralelo, así como un pequeña introducción de los distintos lenguajes y bibliotecas que se usarán. De esta forma se pretende explicar conceptos importantes que servirán para el presente trabajo.

En el capítulo III se da una descripción del método numérico conocido como el Método de la Ecuación Integral (Mendoza-Suárez *et al.*, 2006), para calcular la respuesta óptica mediante la reflectancia y transmitancia de una PCW de conductor real. Además, se combinará con el Método de la Condición a la Frontera de Impedancia y el Modelo de Drude, lo cual nos permitirá modelar al conductor real de una manera más precisa.

En el capítulo IV se desarrollan los algoritmos bajo los paradigmas de OpenMP, MPI, ScaLAPACK y CUDA para calcular la respuesta óptica del sistema de estudio. Además se muestran los resultados numéricos calculando los tiempos de cómputo de los diferentes paradigmas que se consideraron en este trabajo.

[11] Por sus siglas en inglés, Perfectly Mathed Layer.

En el capítulo V se describe un método numérico general para aproximación de soluciones de ecuaciones diferenciales parciales muy utilizado en problemas de ingeniería y física, conocido como el Método de Elementos Finitos, el cual será utilizado en su formulación débil de Galerkin.

En el capítulo VI se presentan los resultados obtenidos al resolver la ecuación de Helmholtz en 1D y 2D utilizando FEM en su formulación débil de Galerkin en combinación con la técnica PML y aplicado a problemas en espacios abiertos.

En el capítulo VII se muestra la intensidad del campo cercano de una PCW de conductor perfecto utilizando el IEM y el FEM.

Finalmente en el capítulo VIII se presentan las conclusiones más importantes sobre los resultados obtenidos en el desarrollo de la tesis de investigación.

Capítulo II

INTRODUCCIÓN A LA PROGRAMACIÓN EN PARALELO

En este capítulo se presentan los conceptos básicos relacionados con la programación en paralelo; así como una pequeña introducción de los distintos lenguajes y bibliotecas que se usan para la programación en paralelo.

II.1. Descripción general en paralelo

La computación en paralelo es una técnica de programación en la que muchas instrucciones se ejecutan simultáneamente. Se basa en el principio de que los problemas grandes se pueden dividir en partes más pequeñas que se pueden resolverse de forma concurrente.

Para poder paralelizar un problema y mejorar el rendimiento, es necesario conocer el ambiente de trabajo y algunos conceptos fundamentales sobre paralelización como son:

- Arquitectura de la computadora (Hardware).
- Soporte del sistema operativo.

- Herramienta de software para la implementación.

En la computación en paralelo, el software y hardware están estrechamente relacionados. A fin de lograr la ejecución paralela del software, el hardware debe proporcionar una plataforma que admita la ejecución simultánea de varios procesos o múltiples subprocesos.

En este capítulo se describen algunos conceptos fundamentales sobre la programación en paralelo y el ambiente de trabajo necesario para llevar a cabo un óptimo desarrollo de un programa en paralelo (Pérez Hernández, 2015; Sánchez López, 2016).

II.1.1. Conceptos y terminologías de la programación en paralelo

- **Tarea**: Sección lógica discreta de trabajo computacional. Suele ser un programa o un conjunto de instrucciones que es ejecutable por un procesador.

- **Tarea en paralelo**: Tarea que puede ser ejecutada de forma segura (produciendo resultados correctos) por múltiples procesadores simultáneamente.

- **Ejecución en serie**: Ejecución de un programa de forma secuencial; es decir, una expresión en cada instante de tiempo. No obstante, todas las tareas paralelas tendrán secciones de un programa paralelo que deben ser ejecutadas en serie.

- **Ejecución en paralelo**: Ejecución de un programa por más de una tarea, siendo cada tarea capaz de ejecutar la misma expresión o una diferente, y todas en un mismo instante de tiempo.

- **Algoritmos paralelos**: Son métodos para resolver problemas computacionales en máquinas paralelas.

- **Memoria compartida**: Desde el punto de vista del hardware, la memoria compartida describe una arquitectura de ordenador, donde todos los procesadores

tienen acceso directo a una memoria física común. Desde el punto de vista del software, describe un modelo donde todas las tareas paralelas tienen la misma representación de la memoria, que pueden direccionar y acceder directamente a las mismas localizaciones de una memoria lógica, sin preocuparse donde se encuentra la memoria física.

- **Memoria distribuida**: En el sentido del hardware, la memoria distribuida se refiere a un acceso a memoria basado en red para una memoria física que no es común. Como modelo de programación, las tareas sólo pueden ver lógicamente la memoria de la máquina local y deben utilizar comunicaciones para acceder a la memoria de otras máquinas donde se ejecutan el resto de tareas.

- **Comunicaciones**: Típicamente, las tareas paralelas necesitan intercambiar datos. Hay varios caminos para realizar esto, tales como tener una memoria compartida en bus o sobre una red. Sin embargo, en la actualidad, el hecho de intercambiar datos se le denomina comunicaciones, independientemente del método empleado.

- **Sincronización**: Es la coordinación de tareas paralelas en tiempo real, a menudo asociado a las comunicaciones. La forma en que se implementa la sincronización entre tareas es poniendo un punto de sincronismo dentro del código de una aplicación, de modo que las tareas no continuarán con su trabajo hasta que todas las tareas hayan llegado al mismo punto de sincronismo. La sincronización implica la espera de, al menos una tarea y, por lo tanto, puede causar el incremento del tiempo de ejecución de la aplicación paralela. El tema del sincronismo es muy importante y es uno de los temas más delicados a tratar a la hora de escribir aplicaciones en paralelo.

- **Granularidad**: En la computación en paralelo, la granularidad es una medida

cualitativa de la tasa de tiempo de computación entre tiempo de comunicaciones. En el límite, granularidad gruesa (fina) es cuando grandes (pequeñas) cantidades de trabajo computacional son realizadas en medio de eventos de comunicaciones.

- **Concurrencia**: Define la ejecución simultánea de dos o más procesos en uno o más procesadores.

- **Escalabilidad**: Capacidad de un sistema paralelo (hardware o software) para demostrar un incremento proporcional en la velocidad de ejecución en paralelo con el aumento del número de procesadores. Un factor importante que influye en la escalabilidad es la forma en que se ha diseñado el código de la aplicación paralela a ejecutar.

- **Rendimiento**: Es la proporción entre el resultado que se obtiene y los medios que se emplearon para alcanzar al mismo. En nuestro caso esto es de suma importancia, debido a que entre más procesadores se involucran en la solución de un determinado problema, mejor será el tiempo de procesamiento, pero también incrementará el tiempo de comunicación entre los procesadores involucrados. En el tiempo de cómputo, Amdahl define los límites de aceleración que una aplicación puede alcanzar (Amdahl, 1967). En el tiempo de comunicación, propone varios modelos, donde algunos estiman el tiempo con un conjunto de parámetros y otros parámetros más detallistas, para los patrones de comunicación punto a punto y colectiva.

Ley de Amdahl

En cualquier programa paralelo es imposible dejar que el código secuencial deje de existir, el cual no puede ser paralelizable, y por lo tanto debe ser ejecutado por un sólo procesador. Si la fracción del problema inherentemente secuencial es f, entonces

el tiempo de ejecución de la parte secuencial del programa es $f \times t_1$ y el tiempo en ejecutar la parte paralelizable en p procesadores es $(1-f) \times t_1/p$. Luego el tiempo de ejecución del programa paralelizado será:

$$t_p = f \times t_1 + \frac{(1-f) \times t_1}{p} \tag{1}$$

y por lo tanto, la aceleración será:

$$S_p = \frac{t_1}{t_p} = \frac{1}{f + \frac{(1-f)}{p}}. \tag{2}$$

De lo cual se observa, que aunque se contara con una infinidad de procesadores, la aceleración estará limitada por $1/f$, como puede verse de la siguiente relación:

$$\lim_{p\to\infty} S_p = \lim_{p\to\infty} \left[\frac{1}{f + \frac{(1-f)}{p}} \right] = \frac{1}{f}. \tag{3}$$

De manera análoga, la eficiencia de un programa se acercaría a cero conforme se aumenta el número de procesadores y el programa paralelizado será más susceptible de perder el rendimiento del sistema,

$$\lim_{p\to\infty} E_p = \lim_{p\to\infty} \left[\frac{S_p}{p} \right] = 0. \tag{4}$$

La mejora en la velocidad de ejecución de un programa como resultado de la paralelización está limitada por la porción del programa que no se puede paralelizar como se muestra en la Fig. 4.

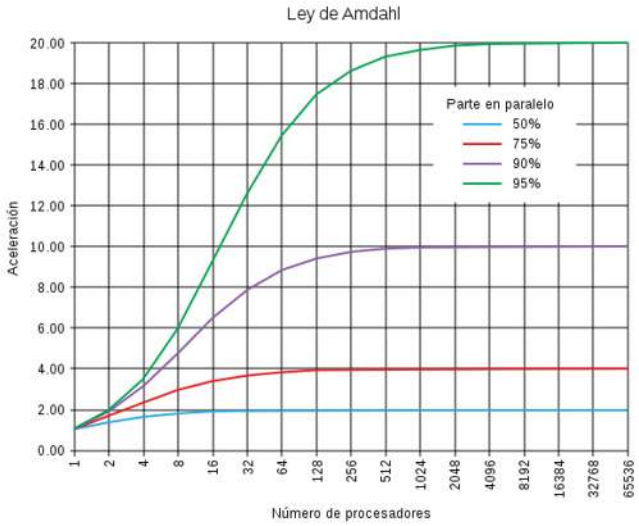

Figura 4: Representación gráfica de la ley de Amdahl.

II.2. Hardware

Existen dos tipos de hardware para llevar a cabo la computación en paralelo. Uno de ellos es la unidad de procesamiento central (CPU[12]) y el otro es la unidad de procesamiento gráfico (GPU[13]). Cada uno con diferentes filosofías de diseño como se muestra en la Fig. 5.

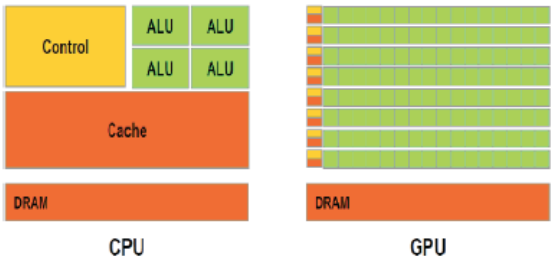

Figura 5: CPU y GPU, diferencia de filosofía de diseño.

[12]Por sus siglas en inglés, Central Processing Unit.
[13]Por sus siglas en inglés, Graphics Processor Unit.

Para más información relacionada con la arquitectura del CPU y GPU consultar la Ref. (Sánchez López, 2016).

II.3. Plataformas de programación en paralelo

Los lenguajes de programación en paralelo, las bibliotecas, las APIs[14] y los modelos de programación en paralelo han sido creados para la programación de computadoras con arquitectura en paralelo. Hay muchos lenguajes y modelos de programación en paralelo. Los que más se utilizan son MPI (Pacheco, 2011) y OpenMP (Chapman *et al.*, 2008) para el caso de CPU, y para GPU existen CUDA (Cook, 2013) y OpenCL (Kaeli *et al.*, 2015). Este último se desarrolló con el fin de poder escribir programas que se puedan ejecutar a través de plataformas heterogéneas.

II.3.1. OpenMP

OpenMP es una API para la programación multiproceso de memoria compartida en múltiples plataformas. Se compone de un conjunto de directivas de compilador, rutinas de biblioteca y variables de entorno que facilitan la descripción de una o más porciones de un programa que deben ser ejecutadas por varias unidades de procesamiento de manera simultánea. Una gran ventaja que ofrece es que de una manera muy simple se pueden paralelizar programas secuenciales sin grandes cambios en el código fuente.

II.3.2. MPI

Interfaz de paso de mensajes (MPI) es un modelo de comunicación ampliamente usado en computación paralela. Una de las ventajas que ofrece es que, la sintaxis se ha estandarizado y cuenta con una amplia cantidad de funciones que se encuentran en su

[14]Por sus siglas en inglés, Application Programming Interface.

biblioteca. Esto garantiza la portabilidad de los programas paralelos; sin embargo, los resultados pueden variar de una implementación a otra. Para más detalles de como implementar programas usando MPI consultar la Ref. (Pérez Hernández, 2015, 30–51).

SCALAPACK

ScaLAPACK (Scalable Linear Algebra PACKage) es una biblioteca de alto rendimiento diseñada para la computación heterogénea. Sus rutinas son un rediseño de LAPACK (Linear Algebra PACKage) para computadoras paralelas de memoria distribuida. Resuelve sistemas lineales densos, problemas de mínimos cuadrados, problemas de valores propios y de valores singulares entre otros, todo esto haciendo uso de la distribución cíclica en bloques y algoritmos de bloque dividido. Esto nos garantiza que el programa se ejecuta tan rápido como sea posible y que funcionará en cualquier máquina donde BLAS, LAPACK y BLACS estén disponibles (Blackford *et al.*, 1997). Para más detalles de como implementar programas usando ScaLAPACK consultar la Ref. (Lozano Trejo, 2017, 14–26).

II.3.3. CUDA

CUDA (Computer Unified Device Architecture) es una plataforma de computación en paralelo que presenta un nuevo modelo de programación y un conjunto de instrucciones, que en conjunto nos permite implementar el paralelismo en el procesamiento de tareas y datos con diferentes niveles de granularidad. El desarrollador puede usar una variación del lenguaje de programación C para codificar los algoritmos. También es compatible con otros lenguajes de programación como: C++, Python, FORTRAN y Java. Para más detalles consultar la Ref. (Sánchez López, 2016, 24–38).

II.3.4. OpenCL

OpenCL (Open Computing Language) es tanto una interfaz de programación de aplicaciones y de un lenguaje de programación, lo cual nos permite escribir programas con paralelismo de datos y de tareas, que se ejecutan a través de plataformas heterogéneas, como son: CPU, GPU, DSP (Digital Signal Processors), FPGAs (Field-Programmable Gate Arrays), entre otros. En la presente tesis no se hará uso de OpenCL ya que por la falta de librerías optimizadas en el área de soluciones de ecuaciones lineales resulta impráctico su uso.

Capítulo III

MÉTODO DE LA ECUACIÓN INTEGRAL

En este capítulo se describe una técnica rigurosa para modelar la interacción de la luz con una guía de onda de cristal fotónico (PCW) finita. La técnica se conoce como el Método de la Ecuación Integral (IEM). En particular, lo aplicaremos para estudiar la propagación de luz a través de una PCW de conductor perfecto calculando el campo cercano a distintas frecuencias (ver capítulo VII). Para el cálculo de la reflectancia y transmitancia como función de la frecuencia para una PCW de conductor real se utilizará el Método de la Condición a la Frontera de Impedancia y el Modelo de Drude para complementar el IEM y poder modelar al conductor real con mayor precisión.

El método numérico para calcular la respuesta óptica de una PCW está basado en la solución numérica de la ecuación de Helmholtz usando ecuaciones integrales (Mendoza-Suárez *et al.*, 2006; Pérez *et al.*, 2009; Mendoza-Suárez and Pérez-Aguilar, 2016). Este método parte del segundo teorema integral de Green permitiendo obtener un par de ecuaciones integrales acopladas que involucran, como incógnitas el campo y su derivada normal evaluados en las fronteras o superficies involucradas. La discretización del sistema resulta en una ecuación matricial inhomogénea cuya solución determina las funciones fuente, con las que se puede calcular el campo cercano, la reflectancia y

transmitancia del sistema de estudio.

Este método implica ecuaciones independientes del tiempo, por lo que la evolución en el tiempo de los sistemas no es una preocupación, y funciona a lo largo de los contornos de las fronteras involucradas en la geometría que se manejan. Esto presenta algunas ventajas en comparación con otros métodos, ya que sólo toma en cuenta un número finito de puntos de muestreo a lo largo de los contornos del sistema, permitiendo una menor cantidad de recursos computacionales.

III.1. Descripción del método de la ecuación integral

A continuación se presenta un planteamiento del IEM para calcular el campo lejano de una PCW bidimensional finita formada por dos paredes internas planas que encierran un arreglo de inclusiones cilíndricas circulares (Mendoza-Suárez and Pérez-Aguilar, 2016).

La geometría del sistema que se consideró para la teoría se muestra en la Fig. 6. Un sistema de q cuerpos invariantes a lo largo de z, es iluminado por un haz Gaussiano. El plano de incidencia es el x-y. La región 0 se caracteriza por un índice real de refracción $n_0 = \sqrt{\varepsilon_0(\omega)}$ y las regiones de la 1 a q se definen por las curvas Γ_j, y se caracterizan por el correspondiente índice de refracción n_j o alternativamente por la constante dieléctrica $\epsilon_j(\omega)$.

De la ecuación de onda que involucra campos armónicos en el tiempo, se obtiene la ecuación de Helmholtz,

$$\nabla^2\psi(\mathbf{r}) + k^2\psi(\mathbf{r}) = 0, \tag{5}$$

donde $\psi(\mathbf{r})$ representa el campo eléctrico o magnético y k el vector de onda de propagación.

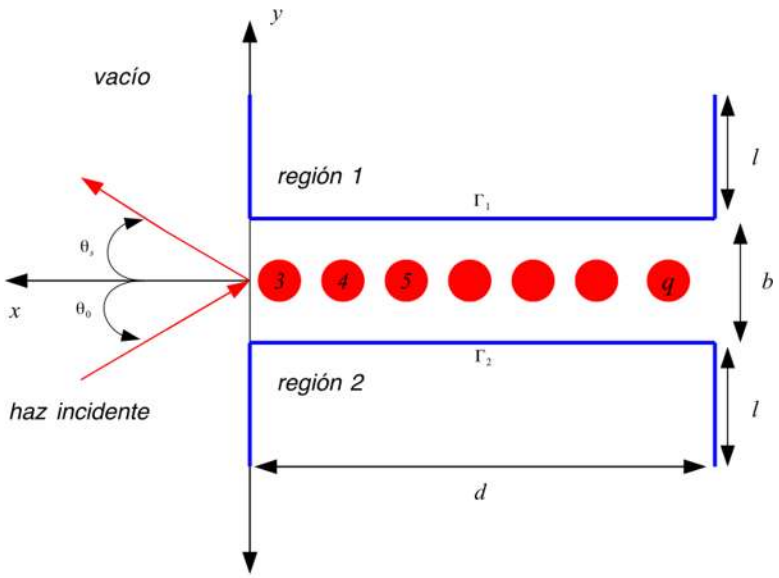

Figura 6: Descripción esquemática de una PCW, de ancho b y longitud d con inclusiones cilíndricas, iluminado por haz Gaussiano en la región del vacío. Las regiones 1 y 2 constituyen las placas paralelas conductoras, mientras que las regiones 3, 4 y así sucesivamente constituyen las inclusiones circulares conductoras.

Así podemos asociar una función de Green para el j-ésimo medio,

$$\nabla^2 G_j(\mathbf{r}, \mathbf{r}') + k_j^2 G_j(\mathbf{r}, \mathbf{r}') = 4\pi\delta(\mathbf{r} - \mathbf{r}'), \tag{6}$$

donde $G_j(\mathbf{r}, \mathbf{r}')$ representa el propagador del campo debido a una fuente luz puntual que emite a la frecuencia ω en la posición $\mathbf{r}'$, correspondiente a cada medio. La $\delta(\mathbf{r} - \mathbf{r}')$ es la delta de Dirac. Una solución de la Ec. (6) está dada por

$$G_j(\mathbf{r}, \mathbf{r}') = i\pi H_0^{(1)}(k_j|\mathbf{r} - \mathbf{r}'|), \tag{7}$$

siendo $H_0^{(1)}(x)$ la función de Hankel de primera especie y orden cero.

Empleando el segundo teorema integral de Green (Jackson, 1999) aplicado a las funciones $\psi(\mathbf{r})$ y $G_j(\mathbf{r}, \mathbf{r}')$ correspondientes a la región 0, obtenemos

$$\psi^{(0)}(\mathbf{r}) = \psi^{(0)}_{inc}(\mathbf{r}) + \frac{1}{4\pi}\sum_{j=1}^{q}\int_{\Gamma_j}\left[\frac{\partial G_j(\mathbf{r},\mathbf{r}')}{\partial n_j}\psi^{(j)}(\mathbf{r}) - G_j(\mathbf{r},\mathbf{r}')\frac{\partial \psi^{(j)}}{\partial n}(\mathbf{r})\right] ds. \qquad (8)$$

En esta expresión, $\psi^{(0)}_{inc}(\mathbf{r})$ representa el campo incidente y la suma de las integrales representan el campo esparcido.

Siguiendo los mismo pasos, para la j-ésima región, el campo $\psi^{(j)}(\mathbf{r})$ puede expresarse como

$$\theta(\mathbf{r})\psi^{(j)}(\mathbf{r}) = \frac{1}{4\pi}\int_{\Gamma_j}\left[\frac{\partial G_j(\mathbf{r},\mathbf{r}')}{\partial n_j}\psi^{(j)}(\mathbf{r}) - G_j(\mathbf{r},\mathbf{r}')\frac{\partial \psi^{(j)}}{\partial n}(\mathbf{r})\right] ds, \qquad (9)$$

donde $\theta(\mathbf{r})$ es la función

$$\theta(\mathbf{r}) = \begin{cases} 1 \ \text{ si } \ \mathbf{r} \in S \\ 0 \ \text{ si } \ \mathbf{r} \notin S, \end{cases} \qquad (10)$$

siendo S una superficie cerrada. Los vectores $\mathbf{r}$ representa el punto de observación (donde se mide el campo eléctrico o magnético) y $\mathbf{r}'$ es la variable de integración o la posición de las fuentes.

Las Ecs. (8) y (9) forman un sistema de ecuaciones integrales con las que se puede obtener el campo total en el medio de incidencia y de esparcimiento. Para calcular el campo esparcido usando el segundo término del lado derecho de la Ec. (8), es necesario encontrar una forma de obtener las funciones fuentes a partir de las ecuaciones integrales. Para esto, se hace una aproximación del punto de observación en la región 0 a la superficie de la región j. Por tanto, se obtienen las siguientes ecuaciones integrales acopladas

$$\psi^{(0)}(\mathbf{r}) \ = \ \psi^{(0)}_{inc}(\mathbf{r}) + \frac{1}{4\pi}\sum_{j=1}^{q}\int_{\Gamma_j}\left[\frac{\partial G_j(\mathbf{r},\mathbf{r}')}{\partial n_j}\psi^{(j)}(\mathbf{r}) - G_j(\mathbf{r},\mathbf{r}')\frac{\partial \psi^{(j)}}{\partial n}(\mathbf{r})\right] ds \qquad (11)$$

$$0 = \frac{1}{4\pi}\int_{\Gamma_j}\left[\frac{\partial G_j(\mathbf{r},\mathbf{r}')}{\partial n_j}\psi^{(j)}(\mathbf{r}) - \frac{f_i}{f_0}G_j(\mathbf{r},\mathbf{r}')\frac{\partial\psi^{(j)}}{\partial n}(\mathbf{r})\right]ds, \tag{12}$$

donde δ_{ij} es la delta de Kroneker y $G_j(\mathbf{r},\mathbf{r}')$ está dada por la Ec. (7), con $i = 1$ hasta q y las expresiones $f_{0,j} = \varepsilon_{0,j}(\omega)$ para la polarización TE y $f_{0,j} = 1, \mu_{0,j}(\omega)$ para la polarización TM, dependiendo de que material sea el j-ésimo medio.

En la Ec. (9) la superficie S_j está limitada por el contorno cerrado Γ_j correspondiente y la derivada normal $\partial/\partial n' \equiv \hat{\mathbf{n}}\cdot\nabla\mathbf{n}$ va hacia afuera del contorno Γ_j.

De la Ec. (9), a cada uno de los términos los etiquetamos como

$$\frac{1}{4\pi}\oint_{\Gamma_j} G(\mathbf{r},\mathbf{r}')\frac{\partial\psi(\mathbf{r})}{\partial n'}ds' \cong \sum_{n=1}^{N_q} L^{(j)}_{mn(q)}\Phi^{(j)}_{n(q)}, \tag{13}$$

$$\frac{1}{4\pi}\oint_{\Gamma_j} \Psi(\mathbf{r})\frac{\partial G(\mathbf{r},\mathbf{r}')}{\partial n'}ds' \cong \sum_{n=1}^{N_q} N^{(j)}_{mn(q)}\Psi^{(j)}_{n(q)}, \tag{14}$$

donde $\Psi^{(j)}_{n(q)} = \psi_j(\mathbf{r}')|_{\mathbf{r}'=\mathbf{r}_{n(q)}}$ es el campo y $\Phi^{(j)}_{n(q)} = \frac{\partial\psi_j(\mathbf{r}')}{\partial n'_j}|_{\mathbf{r}'=\mathbf{r}_{n(q)}}$ su derivada normal, m representa el m-ésimo punto [coordenadas del observador $\mathbf{r} = (x_m, y_m)$] a lo largo del contorno Γ_j de cada región ($q = 1, 2, 3, ...$). Los elementos de matriz están dados por (Mendoza-Suárez *et al.*, 2006)

$$L^{(j)}_{mn} = [1-\delta_{mn}]\frac{i\Delta s}{4}H^1_o(k_jR_{mn}) + \left[\frac{i\Delta s}{4}H^1_o\left(k\frac{\Delta s}{2e}\right)\right]\delta_{mn}, \tag{15}$$

$$N^{(j)}_{mn} = [1-\delta_{mn}]\frac{i\Delta s}{4}k_j\hat{\mathbf{n}}_n\cdot\frac{\mathbf{R}_{mn}}{R_{mn}}H^{(1)}_1(k_jR_{mn}) + \left[\frac{1}{2} + \frac{\Delta s}{4\pi}\hat{\mathbf{n}}_n\cdot\hat{\mathbf{t}}'_n\right]\delta_{mn}, \tag{16}$$

siendo

$$\begin{aligned}\hat{\mathbf{n}}_n\cdot\mathbf{R}_{mn} &= -y'(s)(x_m - x_n) + x'(s)(y_m - y_n),\\ \hat{\mathbf{n}}_n\cdot\hat{\mathbf{t}}'_n &= x'(s)y''(s) - y'(s)x''(s),\\ R_{mn} &= \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2},\end{aligned} \tag{17}$$

y $H_1^{(1)}(x)$ es la función de Hankel de primera especie y primer orden.

III.1.1. El campo y potencia incidente

Para tratar el problema de la propagación de la luz a través de una guía de ondas con el método previamente descrito es necesario hacer ciertas consideraciones. Como el tamaño del sistema es finito, para evitar efectos de borde se ilumina con un haz Gaussiano angosto cuya intersección con el plano de la guía de ondas tiene anchura media g. Este parámetro debe ser menor que la longitud total del sistema $L_y = 2l + b$, pero mucho más grande que la anchura de la abertura b (ver Fig. 6). Con estas consideraciones el campo incidente se puede expresar en términos de su espectro angular,

$$\psi_{inc}(x,y) = \frac{1}{2\pi} \int_{-n_0(\omega/c)}^{n_0(\omega/c)} A(u,k) \exp\left[iux - \alpha_0(u)y\right] du, \tag{18}$$

donde $\alpha_0(u) = \left[\left(n_0\left(\omega/c\right)\right)^2 - u^2\right]^{1/2}$, con $\mathrm{Re}\left\{\alpha_0(u)\right\} > 0$ y $\mathrm{Im}\left\{\alpha_0(u)\right\} > 0$. En nuestro caso elegimos

$$A(u,k) = \psi_0 \sqrt{\pi} g \exp\left[-g^2(u-k)/4 + i\alpha_0(u)d\right], \tag{19}$$

siendo ψ_0 una constante con unidades apropiadas.

Con este campo incidente podemos encontrar la potencia incidente total a través del plano $L_y L_z$, donde L_z es la longitud en la dirección z. Para esto, se emplea el vector de Poynting $\mathbf{S}$, que proporciona la dirección y la magnitud del flujo de energía por unidad de tiempo, que está dado por

$$S = \frac{c}{8\pi} \mathbf{E} \times \mathbf{H}^*. \tag{20}$$

La parte real de esta expresión proporciona una medida de la irradiancia, o flujo de energía promedio por unidad de tiempo. Para el caso de polarización TE, de la ecuación

anterior se tiene que la componente del vector de Poyting a lo largo del eje y es

$$S_y = \frac{c}{8\pi}\mathrm{Re}\left\{H_x^* E_z\right\}, \tag{21}$$

o bien, en términos del campo eléctrico,

$$S_y = \frac{c^2}{8\pi\omega}\mathrm{Re}\,\partial E_z\left\{-iE_z\frac{\partial E_z}{\partial y}\right\}. \tag{22}$$

Para el caso de la polarización TM, la componente del vector de Poyting es

$$S_y = \frac{c^2}{8\pi\omega\varepsilon}\mathrm{Re}\left\{-iH_z^*\frac{\partial H_z}{\partial y}\right\}. \tag{23}$$

Por consiguiente, la potencia incidente total es

$$P_{inc}(k) = L_z\sqrt{\frac{\pi}{2}}g\alpha_0(k)\frac{c^2}{8\pi\varepsilon\omega}\left|\psi_0\right|^2, \tag{24}$$

donde se ha supuesto que $(\omega/c) \gg 1$.

III.1.2. El campo y potencia esparcida

El campo esparcido está representado por el segundo término del lado derecho de la Ec. (12). Partiendo de la ecuación se puede obtener una expresión para los campos reflejado y transmitido del espectro angular. Para esto, se usa una expansión en términos de ondas planas para la función de Green definida como

$$G_0(\mathbf{r},\mathbf{r}') = \int_{-\infty}^{\infty}\frac{du}{2\pi\alpha_0(u)}\exp\left\{iu(x-x') + i\alpha_0(u)\left|y-y'\right|\right\}, \tag{25}$$

donde $\mathbf{r}' = (\xi(s),\eta(s))$ que nos permiten encontrar el campo esparcido de la forma

$$\psi_{sc}^{(0)}(\mathbf{r}) = \int_{-\infty}^{\infty} \frac{du}{2\pi} \frac{1}{4\pi} \sum_{j=1}^{q} \int_{\Gamma_j} \left[\frac{2\pi i}{\alpha_0(u)} \left[-i\mathbf{n} \cdot \mathbf{U} \right] \psi^{(0)}(s) - \frac{2\pi i}{\alpha_0(u)} \frac{\partial \psi^{(0)}(s)}{\partial n} \right] \times \exp\left\{ iu(x - \xi(s)) + i\alpha_0(u) \left| y - \eta(s) \right| \right\} ds, \tag{26}$$

donde se ha definido como $\mathbf{U} = (u \pm \alpha_0(u))$.

Considerando nuestra geometría y la Ec. 26, para el espacio $x > d$, se tiene que el campo reflejado es

$$\psi_{sc}^{(0)+}(x, y) = \int_{-\infty}^{\infty} \frac{du}{2\pi} S^{+}(u, k) \exp\left\{ iux + i\alpha_0(u) y \right\}, \tag{27}$$

donde

$$S^{+}(u, k) = \frac{-i}{2\alpha_0(u)} \sum_{j=1}^{q} \int_{\Gamma_j} \left[i\mathbf{n} \cdot \mathbf{U} \psi_0(s) + \frac{\partial \psi_0(s)}{\partial n} \right] \exp\left\{ -i \left[u\xi(s) + \alpha_0(u)\eta(s) \right] \right\} ds \tag{28}$$

y para el espacio $x < 0$, el campo transmitido tiene la forma

$$\psi_{sc}^{(0)-}(x, y) = \int_{-\infty}^{\infty} \frac{du}{2\pi} S^{-}(u, k) \exp\left\{ iux - i\alpha_0(u) y \right\}, \tag{29}$$

con

$$S^{-}(u, k) = \frac{-i}{2\alpha_0(u)} \sum_{j=1}^{q} \int_{\Gamma_j} \left[i\mathbf{n} \cdot \mathbf{U} \psi_0(s) + \frac{\partial \psi_0(s)}{\partial n} \right] \exp\left\{ -i \left[u\xi(s) - \alpha_0(u)\eta(s) \right] \right\} ds. \tag{30}$$

De aquí que el campo total, para la región $x < 0$ sea

$$\psi_{tot}^{(0)-}(x, y) = \int_{-\infty}^{\infty} \frac{du}{2\pi} \left[A(u, k) + S^{-}(u, k) \right] \exp\left\{ iux - i\alpha_0(u) y \right\}, \tag{31}$$

y para la región $x > d$ de la forma

$$\psi_{tot}^{(0)+}(x,y) = \int_{-\infty}^{\infty} \frac{du}{2\pi} S^{+}(u,k) \exp\left\{iux + i\alpha_0(u)y\right\}, \tag{32}$$

donde

$$\begin{aligned} S^{\pm} &= \frac{i}{2\alpha_0(u)} \sum_{j=1}^{q} \int_{\Gamma_j} \left\{ i\left[u\eta'(s) \mp \alpha_0(u)\xi'(s)\right]\psi_0(s) - \frac{\partial\psi_0(s)}{\partial n} \right\} \\ &\quad \times \exp\left\{-i\left[u\xi(s) \pm \alpha_0(u)\eta(s)\right]\right\} ds \end{aligned} \tag{33}$$

representa el espectro angular del campo esparcido.

Para el caso de transmisión, $S^{-}(u,k)$ y $u < n_0(\omega/c)$, las componentes del vector de onda son $u = n_0(\omega/c)\sin\theta_t$ y $\alpha_0(u) = n_0(\omega/c)\cos\theta_t$, mientras que para el caso de reflexión, $S^{+}(u,k)$ y $u < n_0(\omega/c)$, $u = n_0(\omega/c)\sin\theta_r$ y $\alpha_0(u) = n_0(\omega/c)\cos\theta_r$ (ver Fig. 6).

Haciendo uso de las Ecs. (22), (23), (27) y (29), se obtiene la expresión para la *potencia esparcida*, en términos del espectro angular,

$$\begin{aligned} P_{sc}^{\pm}(k) &= \pm L_z \frac{c^2}{8\pi\omega\varepsilon} \int dx\, \mathrm{Re} \int_{-n_0(\omega/c)}^{n_0(\omega/c)} \frac{du}{2\pi} \int_{-n_0(\omega/c)}^{n_0(\omega/c)} \frac{du'}{2\pi} \alpha_0(u) S^{\pm}(u,k) S^{\pm *}(u,k)(u',k) \\ &\quad \times \exp\left\{i(u-u')x \pm i(\alpha_0(u) - \alpha_0^{*}(u'))y\right\} \\ &= \pm L_z \frac{c^2}{8\pi\omega\varepsilon} \int_{-n_0(\omega/c)}^{n_0(\omega/c)} \frac{du}{2\pi} \alpha_0(u) \left|S^{\pm}(u,k)\right|^2 . \end{aligned} \tag{34}$$

Ahora, utilizamos las Ecs. (24) y (34), el coeficiente de reflexión diferencial (DRC[15]) puede ser expresado como

[15]Por sus siglas en inglés, Differential Reflection Coefficient.

$$\left(\frac{\partial R}{\partial k}\right) = \frac{P_{sc}^{\pm}(k)}{P_{inc}(k_y)} = \pm\frac{1}{F(k)} \int_{-n_0(\omega/c)}^{n_0(\omega/c)} \frac{du}{2\pi}\alpha_0(u)\left|S^{\pm}(u,k)\right|^2, \tag{35}$$

donde

$$F(k) = \sqrt{\frac{\pi}{2}} g\alpha_0(k)\left|\psi_0\right|^2. \tag{36}$$

Finalmente utilizando las Ecs. (24) y (34), se obtiene la Reflectancia R y la Transmitancia T respectivamente, como:

$$R(k) = \frac{P_{sc}^{+}(k)}{P_{inc}(k_y)} = \frac{1}{F(k)} \int_{-n_0(\omega/c)}^{n_0(\omega/c)} \frac{du}{2\pi}\alpha_0(u)\left|S^{+}(u,k)\right|^2, \tag{37}$$

$$T(k) = \frac{P_{sc}^{-}(k)}{P_{inc}(k_y)} = \frac{1}{F(k)} \int_{-n_0(\omega/c)}^{n_0(\omega/c)} \frac{du}{2\pi}\alpha_0(u)\left|S^{-}(u,k)\right|^2. \tag{38}$$

Es importante mencionar, que el campo incidente aparece en la Ec. (38) por que el campo total en la región 0 es el resultado de la interferencia de los campos incidente y esparcido. Para el balance de la energía se debe tener que $R(k) + T(k) \leq 1$, donde la igualdad se debe cumplir para el caso de conductores perfectos.

III.2. Método de la Condición a la Frontera de Impedancia

Se presenta una condición a la frontera de impedancia (IBCM[16]) para superficies unidimensionales, la cual permite el estudio de la difracción y esparcimiento de ondas electromagnéticas por una superficie conductora. El problema teórico es simplificado notablemente si la superficie puede ser considerada como un material perfectamente

[16]Por sus siglas en inglés, Impedance Boundary Condition Method.

conductor. En este caso, los campos no pueden penetrar a la superficie, y solamente el campo electromagnético en el medio incidente necesita ser tomado en consideración (Mendoza-Suárez, 1996).

El uso de una condición a la frontera de impedancia elimina la necesidad de tomar en consideración el campo dentro de la superficie esparcidora, y retiene algunos de los aspectos físicos importantes del problema. En el contexto de cálculos de esparcimiento, es conocido que la condición a la frontera de impedancia representa una buena aproximación, para el caso de metales altamente reflejantes (Mendoza-Suárez, 1996). Otra motivación importante para obtener una expresión para la condición a la frontera de impedancia, es que puede ser utilizada para resolver problemas que se presentan en simulaciones numéricas de esparcimiento de luz, cuando la frecuencia de la luz incidente está en el infrarrojo. La explicación de esos problemas, supone que algunos argumentos de las funciones de Hankel involucradas en el uso del Método Integral, son suficientemente grandes como para provocar problemas en la implementación numérica utilizada. Se puede tratar de resolver este problema aumentando el número de puntos de muestreo, para así tener una mayor precisión en los resultados numéricos. De todas formas, este tipo de solución aumenta significativamente el tiempo de cómputo. El uso de una condición a la frontera de impedancia puede eliminar algunas dificultades, porque precisamente esas funciones de Hankel problemáticas, no intervienen en los cálculos que emplean esta aproximación (Mendoza-Suárez, 1996).

Aunado a lo anterior, tenemos que para realizar simulaciones numéricas con el Método Integral requerimos tener una alta capacidad, debido a que se tiene un número muy grande de ecuaciones a resolver. Es por esto que la condición a la frontera de impedancia nos da algunas ventajas. Esta ventaja viene dada por el hecho de que el número de ecuaciones a resolver disminuye en forma importante (hasta en un 50 %), en donde es aplicable esta condición (Mendoza-Suárez, 1996).

La función de impedancia local, $Z(r)$, está definida por la relación (Jackson, 1999)

$$H_t(\mathbf{r}) = \frac{E_t(\mathbf{r})}{Z(\mathbf{r})}, \tag{39}$$

donde $E_t(\mathbf{r})$ y $H_t(\mathbf{r})$ son las componentes tangenciales de los campos eléctricos y magnético, respectivamente. Aplicando las ecuaciones de Maxwell, las condiciones de frontera y debido a la geometría del sistema tenemos

$$\Phi(\mathbf{r}) = K(\mathbf{r})\Psi(\mathbf{r}), \tag{40}$$

donde Ψ y Φ son el campo y su derivada normal, respectivamente. Con la finalidad de encontrar a la denominada función $K(\mathbf{R})$, como una serie de potencias en términos de la profundidad de piel d de un metal, se define la siguiente transformación de coordenadas:

$$\begin{aligned} x &= \xi(s) - ud\eta'(s), \\ y &= \eta(s) - ud\xi'(s), \end{aligned} \tag{41}$$

entre las variables (s, u) y (x, y). En las dos ecuaciones anteriores podemos observar que el parámetro d es importante en la transformación. La transformación mostrada en la Ec. (41) proviene de representar el punto de observación $\mathbf{r}$ como una suma del vector $\mathbf{R}$ que localiza al punto P del perfil con el vector $ud\widehat{\mathbf{n}}(\mathbf{R})$, que es proporcional al vector unitario $\widehat{\mathbf{n}}(\mathbf{R})$. Esto es

$$\mathbf{r} = \mathbf{R} + ud\widehat{\mathbf{n}}(\mathbf{R}). \tag{42}$$

De la ecuación anterior podemos observar que ud mide el grado de alejamiento entre el punto de observación y la superficie. Con estas consideraciones, damos los primeros

cinco términos de la serie de potencias que representa la función de impedancia:

$$\begin{aligned} K(s) &= (d)^{-1} + \tfrac{1}{2\xi}{}' \left[\eta''\right](d)^0 + \tfrac{1}{8(\xi')^2}\left[(\eta'')^2\right](d)^1 \\ &+ \tfrac{1}{8(\xi')^3}\left[(\xi')^2\eta''' + 3\eta'\eta''\eta''' + 2\left(\eta''\right)^3 + 3\left(\xi''\right)\eta''\right](d)^2 + ... \end{aligned} \tag{43}$$

la cual es utilizada, en puntos en donde $\xi(x) \neq 0$. En aquellos puntos en donde $\xi(x) = 0$ se tiene la expresión

$$\begin{aligned} K(s) &= (d)^{-1} + \tfrac{1}{2\eta}{}' \left[\xi\eta''\right](d)^0 + \tfrac{1}{8(\eta')^2}\left[(\xi'')^2\right](d)^1 \\ &+ \tfrac{1}{8(\eta')^3}\left[(\eta')^2\xi''' + 3\xi'\xi''\xi''' + 2\left(\xi''\right)^3 + 3\left(\eta''\right)\xi''\right](d)^2 + ... \end{aligned} \tag{44}$$

Las dos expresiones anteriores, pueden servir igualmente en puntos donde $\xi(x) \neq 0$ y $\eta(x) \neq 0$. Así, esta condición del IBCM nos permite sustituir el campo por su derivada (o viceversa) para una u otra polarización, ya que podemos modelar el esparcimiento de ondas electromagnéticas por una superficie conductora con una matriz resultante ya no de rango $2M \times 2M$ si no de $M \times M$, reduciendo significativamente el tiempo de cómputo así como la memoria utilizada para modelar los problemas.

III.3. Modelo de Drude

El modelo de Drude o de Lorentz-Drude para la conducción eléctrica fue desarrollado por Paul Drude para explicar las propiedades de transporte de electrones en materiales (especialmente en metales). En este modelo, existe una frecuencia crítica llamada frecuencia de plasma, por debajo de la cual la permitividad eléctrica es negativa y en consecuencia la propagación de ondas electromagnéticas está prohibida, por lo que en metales como el oro y la plata a frecuencias ópticas e infrarrojas, el campo sólo puede penetrar una pequeña distancia. Por encima de la frecuencia de plasma la permitividad es positiva, el medio es transparente y permite la propagación de las

ondas electromagnéticas.

Para analizar los efectos de la luz en medios metálicos, no podemos simplemente hacer uso de $\mathbf{J} = \sigma\mathbf{E}$ (Ley de Ohm) para la densidad de corriente donde σ es la conductividad estática. Es necesario considerar el movimiento real de los electrones bajo la acción del campo eléctrico alterno de la onda luminosa. Para esto el modelo de Drude supone que un portador promedio de carga eléctrica está sujeto a la acción de una "fuerza de resistencia" γ. En presencia de un campo eléctrico externo $\mathbf{E}$ se satisface la siguiente ecuación diferencial:

$$m\frac{d\mathbf{v}}{dt} + m\tau^{-1}\mathbf{v} = -e\mathbf{E}, \tag{45}$$

donde $\mathbf{v}$ es la velocidad del electrón, $\tau = 1/\gamma \approx 10^{-4}$ s es el tiempo de relajación y m es la masa del electrón. Dado que la densidad de corriente es

$$\mathbf{J} = -Ne\mathbf{v}, \tag{46}$$

siendo N el número de electrones de conducción por unidad de volumen o densidad electrónica. Entonces la Ec. (45) se puede expresar en términos de $\mathbf{J}$ como

$$\frac{d\mathbf{J}}{dt} + \gamma\mathbf{J} = \frac{Ne^2}{m}\mathbf{E}. \tag{47}$$

Considerando que el campo eléctrico aplicado y la densidad de corriente de conducción están dadas por

$$\mathbf{E} = \mathbf{E}_0 e^{-i\omega t} \quad \text{y} \quad \mathbf{J} = \mathbf{J}_0 e^{-i\omega t}, \tag{48}$$

que remplazando en la ecuación de movimiento (47) obtenemos

$$\frac{d\left[\mathbf{J}_0 e^{-i\omega t}\right]}{dt} + \gamma \mathbf{J}_0 e^{-i\omega t} = -i\omega \mathbf{J}_0 e^{-i\omega t} + \gamma \mathbf{J}_0 e^{-i\omega t} = \frac{Ne^2}{m}\mathbf{E}_0 e^{-i\omega t} \tag{49}$$

y multiplicando por $e^{i\omega t}$ obtenemos

$$(-i\omega + \gamma)\mathbf{J} = \frac{Ne^2}{m}\mathbf{E}. \tag{50}$$

Para el caso de campos estáticos, $\omega = 0$, se tiene que

$$\mathbf{J} = \frac{Ne^2}{m\gamma}\mathbf{E} = \sigma \mathbf{E}, \tag{51}$$

por lo que,

$$\sigma = \frac{Ne^2}{m\gamma}, \tag{52}$$

que es la conductividad estática. Para el caso general de un campo oscilante aplicado

$$\mathbf{J} = \left[\frac{\sigma}{1 - \frac{i\omega}{\gamma}}\right]\mathbf{E} = \sigma_\omega \mathbf{E}, \tag{53}$$

obtenemos

$$\sigma = \frac{\sigma}{1 - \frac{i\omega}{\gamma}}, \tag{54}$$

que es la conductividad dinámica.

A frecuencias muy bajas $(\omega/\gamma) \ll 1$, la conductividad dinámica es puramente real y los electrones siguen el campo eléctrico. Conforme el campo aplicado va incrementando, la inercia de electrones introduce un retardo de fase en la respuesta de los electrones en el campo y la conductividad dinámica es compleja. Para frecuencias muy altas $(\omega/\gamma) \gg 1$, $\mathbf{J} \approx i\sigma\mathbf{E} = \left(e^{i\frac{\pi}{2}}\right)\sigma\mathbf{E}$ y la conductividad dinámica σ_ω es puramente imaginaria y las oscilaciones de los electrones tienen un desfasamiento de $\pi/2$ con el campo aplicado.

La ecuación de onda para el campo eléctrico es

$$\nabla \times (\nabla \times \mathbf{E}) + \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} = -\mu_0 \frac{\partial^2 \mathbf{P}}{\partial t^2} - \mu_0 \frac{\partial \mathbf{J}}{\partial t}. \tag{55}$$

Para la propagación de ondas electromagnéticas en medios metálicos haremos uso de la expresión dada en la Ec. (53) y que $c^2 = 1/\epsilon_0\mu_0$, por lo que en la Ec. (55) se encuentra la ecuación de onda para metales,

$$\nabla^2 \mathbf{E} = \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} + \frac{1}{\epsilon_0 c^2}\left[\frac{\sigma}{1 - \frac{i\omega}{\gamma}}\right]\frac{\partial \mathbf{E}}{\partial t}. \tag{56}$$

Para resolver la ecuación anterior, se propone un campo eléctrico oscilante, $\mathbf{E} = \mathbf{E}_0 e^{i(\mathbf{k}\cdot\mathbf{r} - \boldsymbol{\omega}\mathbf{t})}$, que representa una onda. Sustituyendo directamente en la Ec. (56) se puede mostrar que ésta es una solución, siempre que

$$k^2 = \frac{\omega^2}{c^2} + i\left[\frac{\omega\mu_0\sigma}{1 - \frac{i\omega}{\gamma}}\right], \tag{57}$$

donde $c^2 = 1/\epsilon_0\mu_0$. Además por la forma de la ecuación anterior se puede considerar que $k = k(\omega)$ como una función de la frecuencia ω. Podemos representar k como

$$k(\omega) = k_R(\omega) + ik_I(\omega). \tag{58}$$

Esto es equivalente a introducir un índice de refracción complejo

$$n(\omega) = n_R(\omega) + in_I(\omega), \tag{59}$$

que en términos de la relación de dispersión es

$$k(\omega) = \frac{\omega}{c} n(\omega). \tag{60}$$

Índice de refracción en un metal

Para calcular el índice de refracción de un metal necesitamos considerar el caso general descrito en la Ec. (57),

$$n^2{=}\frac{c}{\omega}k^2 = 1 + i\left[\frac{c^2\mu_0\sigma}{\omega\left(1 - i\frac{\omega}{\gamma}\right)}\right] = 1 - \frac{c^2\mu_0\sigma\gamma}{\omega^2 + i\omega\gamma}, \tag{61}$$

de donde se define la *"frecuencia de plasma"* como:

$$\omega_p^2 = c^2\mu_0\sigma\gamma = \gamma\left(\frac{Ne^2}{m\gamma}\right)c^2\mu_0 = \frac{Ne^2}{c\epsilon_0}. \tag{62}$$

Por consiguiente el índice de refracción del medio conductor está dado por

$$n^2{=}1 - \left[\frac{\omega_p}{\omega^2 + i\omega\gamma}\right], \tag{63}$$

donde $\omega_p = \sqrt{\frac{Ne^2}{m\epsilon_0}}$ es la frecuencia de plasma.

Los electrones en un plasma son desplazados de un fondo uniforme de iones y los campos eléctricos serán construidos en una dirección para restaurar la neutralidad del plasma regresando de vuelta a los electrones a sus posiciones originales. Debido a su inercia, los electrones sobrepasan la oscilación alrededor de sus posiciones de equilibrio con una frecuencia característica, dicha frecuencia es conocida como frecuencia de plasma. Así mismo, existe una longitud de onda llamada crítica λ_c (o longitud de onda de oscilación de plasma), que es aquella en la que por debajo de ella los metales alcalinos se vuelven transparentes y por encima se vuelven opacos y altamente reflectantes. Esta longitud de onda está dada por

$$\lambda_c{=}\lambda_p = \frac{2\pi c}{\omega_p}. \tag{64}$$

Analizando la expresión del índice de refracción para frecuencias altas ($\omega \gg \gamma$), se

tiene que

$$n^2 \approx 1 - \frac{\omega_p^2}{\omega^2}, \tag{65}$$

donde se ha despreciado el término de γ.

Profundidad de Piel (Skin Depth)

Para el caso en el que k^2 es suficientemente pequeño, es decir

$$k^2 = \frac{\omega^2}{c^2} + i\left[\frac{\omega\mu_0\sigma}{1-\frac{i\omega}{\gamma}}\right] \cong i\omega\mu_0\sigma = \exp\left(i\frac{\pi}{2}\right)\omega\mu_0\sigma. \tag{66}$$

Entonces,

$$\begin{aligned} k &\cong \sqrt{\exp\left(i\tfrac{\pi}{2}\right)\omega\mu_0\sigma} = \exp\left(i\tfrac{\pi}{4}\right)\sqrt{\omega\mu_0\sigma} \\ &= \left[\cos\left(\tfrac{\pi}{4}\right) + i\sin\left(\tfrac{\pi}{4}\right)\right]\sqrt{\omega\mu_0\sigma} = (1+i)\sqrt{\tfrac{\omega\mu_0\sigma}{2}}. \end{aligned} \tag{67}$$

Por lo tanto, la parte real e imaginaria de k son iguales,

$$k_R = k_I = \sqrt{\frac{\omega\mu_0\sigma}{2}}, \tag{68}$$

implicando que

$$n_R = n_I = \left[\frac{c}{\omega}\right] = \sqrt{\frac{c^2\mu_0\sigma}{2\omega}} = \sqrt{\frac{\sigma}{2\omega\epsilon_0}}. \tag{69}$$

En un metal, para una onda que se propaga en la dirección z tenemos que

$$\begin{aligned} \mathbf{E} &= \mathbf{E}_0\exp(ikz) = \mathbf{E}_0\exp(-ik_Iz)\exp\left[i\left(k_Rz-\omega t\right)\right] \\ &= \mathbf{E}_0\exp\left(-\tfrac{z}{d}\right)\exp\left[i\left(k_Rz-\omega t\right)\right], \end{aligned} \tag{70}$$

para el cual la profundidad de piel está dada por

$$d = \frac{1}{k_I} = \sqrt{\frac{2}{\omega\mu_{0\sigma}}} = \sqrt{\frac{2\epsilon_0c^2}{\omega\sigma}}. \tag{71}$$

Ejemplo: la profundidad de piel para la Plata que tiene una conductividad estática $\sigma = 1.586 \times 10^{-8}\ \Omega^{-1}\mathrm{m}^{-1}$ es $d_{silver} = 63.3835\ \mu$m.

III.4. Verificación del método numérico

Por simplicidad en este trabajo únicamente se presenta el caso de la polarización TE; es decir $\psi^{(0)} = 0$. En esta sección se utilizará una PCW de longitud $d = 10\pi$ y de ancho $b = \pi$ con cinco inclusiones circulares de radio $r = 0.177$ (ver Fig. 7(a)). En la Fig. 7(b) se muestra la reflectancia en función de la frecuencia reducida ω_r de una PCW de conductor perfecto y conductor real. El conductor elegido fue la plata (Ag) ya que es un buen conductor, por lo que utilizamos los parámetros de frecuencia de plasma reducida, $\omega_p = 45.6578$ y el parámetro de amortiguación $\gamma = 0.0911$, los cuales se utilizaron para calcular la función dieléctrica de acuerdo al DM.

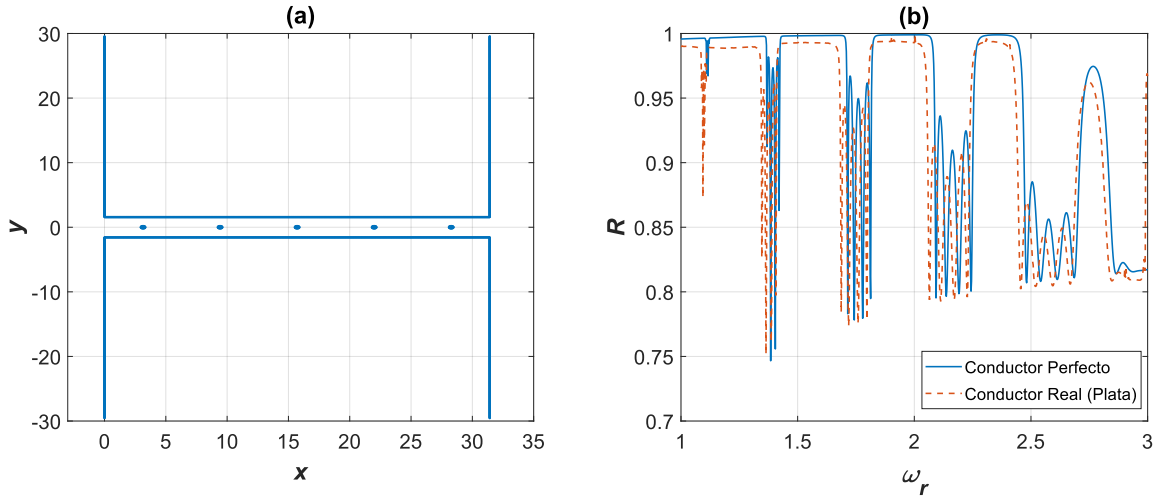

Figura 7: (a) Perfil de la PCW de longitud $d = 10\pi$ periodos y de ancho $b = \pi$ con inclusiones circulares de radio $r = 0.177$. (b) Reflectancia en función de la frecuencia reducida ω_r de una PCW de conductor perfecto (línea continua) y de conductor real (línea punteada).

El resultado numérico del conductor real muestra una buena correspondencia con

el caso ideal que ya se ha probado (Lozano Trejo, 2017; Mendoza-Suárez and Pérez-Aguilar, 2016), lo cual da validez al método integral. Cabe mencionar que el tiempo de cómputo requerido fue de 15223 s, por lo que en el siguiente capítulo se empleará la programación en paralelo para reducir el tiempo considerablemente.

Capítulo IV

RESPUESTA ÓPTICA DE UNA PCW UTILIZANDO EL MÉTODO DE LA ECUACIÓN INTEGRAL

En este capítulo se utilizará el Método de la Ecuación Integral para analizar el tiempo de cómputo requerido para calcular la respuesta óptica de una PCW bidimensional finita de conductor real variando el número de periodos y usando distintos modelos de programación en paralelo.

IV.1. Algoritmos de programación en paralelo

En esta sección mostraremos las diferentes formas de programación en paralelo del Método Integral, usando tanto procesadores como la tarjeta gráfica; así como una combinación de ScaLAPACK y CUDA.

IV.1.1. Algoritmo de MPI

En esta sección se exhibe la implementación de un programa en FORTRAN que utiliza la librería de MPI (Colectiva) para implementar el IEM en forma paralela y poder estudiar la propagación de la luz a través de una PCW de conductor real. En el

programa 1 se muestra el algoritmo que se utilizó para dicho fin. Para conocer el funcionamiento del programa 1 revisar la Ref. (Lozano Trejo, 2017, 45–49). Por tal motivo, no se dará una explicación detallada del algoritmo.

****

Programa 1. Método Integral para una PCW de conductor real usando MPI Colectiva

****

```
PROGRAM PCW_REAL_COLECTIVA
   IMPLICIT NONE
   INCLUDE 'MPIF.H'
   INTEGER, PARAMETER :: MASTER = 0
   INTEGER :: NUMTASKS, TASKID, LEN, IERR, NPROCES, OFFSET
   CHARACTER ( MPI_MAX_PROCESSOR_NAME ) :: HOSTNAME
   INTEGER :: PARTNER, MESSAGE, STATUS ( MPI_STATUS_SIZE )
   INTEGER :: N, INICIO, EPP, EXTRA, J, COLS, CHUNKSIZE, ERROR
   INTEGER NPT, NANG, NPA, NPB, NFF, NW, NPCILI, NPER, POL, FDO
   REAL PI, TINID, DTHET, DELTA, NWA, NWB, LAMBD, B, PHASE, PERIOD, F
   REAL R, P, WNMIN, WNMAX, NR, NI

   PARAMETER ( PI = 3.14159265358979323846264338327 95
   PARAMETER ( B = PI, PHASE = 0.5*PI, PERIOD = 2.0*PI )
   PARAMETER ( LAMBD = 1.033, NFF = 181, NWA = 28.0, NPER = 11 )
   PARAMETER ( NWB = NPER*PERIOD, DELTA = LAMBD/20.0, NANG = 1 )
   PARAMETER ( NPA = NINT ( NWA/DELTA ), F = 0.005 )
   PARAMETER ( R = SQRT ( F*PERIOD*B/PI ), P = 2.0*PI*R )
   PARAMETER ( NPCILI = INT ( P/DELTA ), NPB = NINT ( NWB/DELTA ) )
   PARAMETER ( NPT = 2*( 2*NPA + NPB ) + NPER*NPCILI )
   PARAMETER ( WNMIN = 1.0, WNMAX = 3.0, NW = 200, POL = 2, FDO=2 )

   INTEGER ID, K0, I, IWA ( NPT ), IER
   REAL RAD, G, FK0, DW, EPSI, THETA0 ( NANG ), NORFAC ( NANG )
   REAL SLITW, HSLITW, HLARGE, LARGE, D, A1, A2
   REAL SF ( NPT ), SG ( NPT ), DSF ( NPT ), DSG ( NPT )
   REAL DDSF ( NPT ), DDSG ( NPT )
   REAL CONSUP, CONSDOWN, ENERBAL, REFLPOW, TRANSPOW
   REAL VC, OMWGA_P, GAMMA, OMEGA
   COMPLEX NC, EPS, EI ( NPT, NANG ), ME ( NPT, NPT )
   COMPLEX AK ( NFF, NANG ), EINC ( NPT, NANG ), CIMP ( NPT )
   REAL JIL ( NPT ), JI2 ( NPT ), ETAL ( NPT ), ETA2 ( NPT )
```

```
REAL, DIMENSION ( : ), ALLOCATABLE :: WNUM1, SALIDA1, SALIDA2
REAL, DIMENSION ( : ), ALLOCATABLE :: SALIDA3, SALIDA4
REAL, DIMENSION ( : ), ALLOCATABLE :: WNUM, REFLECT
REAL, DIMENSION ( : ) TRANS, ENERG, FRECUENCY

CALL MPI_INIT (IERR)
CALL MPI_COMM_SIZE ( MPI_COMM_WORLD, NUMTASKS, IERR )
CALL MPI_COMM_RANK ( MPI_COMM_WORLD, TASKID, IERR )
CALL MPI_GET_PROCESSOR_NAME ( HOSTNAME, LEN, IERR )
CHUNSKSIZE = NW/NUMTASKS

ALLOCATE (WNUM ( CHUNKSIZE ), STAT = ERROR )
ALLOCATE ( REFLECT ( CHUNKSIZE ), STAT = ERROR )
ALLOCATE ( TRANS ( CHUNKSIZE ), STAT = ERROR )
ALLOCATE ( ENERG ( CHUNKSIZE ), STAT = ERROR )
ALLOCATE ( FRECUENCY ( CHUNKSIZE ), STAT = ERROR

IF ( TASKID == MASTER ) THEN
   ALLOCATE ( WNUM1 ( NW ) , STAT = ERROR )
   ALLOCATE ( SALIDA1 ( NW ), STAT = ERROR )
   ALLOCATE ( SALIDA2 ( NW ), STAT = ERROR )
   ALLOCATE ( SALIDA3 ( NW ), STAT = ERROR )
   ALLOCATE ( SALIDA4 ( NW ), STAT = ERROR )
ENDIF

IF ( CHUNKSIZE*NUMTASKS . NE . NW ) THEN
   CALL MPI_ABORT ( MPI_COMM_WORLD, IERR)
   STOP
END IF

NC = CMPLX ( NR , NI )
EPSR = NR**2-NI**2
EPSI = 2.0*NR*NI
EPS = CMPLX ( EPSR, EPSI )

IF ( TASKID == MASTER ) THEN
   PRINT*, ' NPT=', NPT
END IF

TINID = 0.0
RAD = PI/180
D = NWB
SLITW = B
HSLITW = 0.5*B
HLARGE = NWA + HSLITW
```

```
 LARGE = 2.0*HLARGE
 A1 = 0.0*B
 A2 = A1
 DTHET = 1.0
 WNUM1 = WNMIN
 G = LARGE/5.0

 DO K0 = 1, NANG
    FK0 = FLOAT ( K0 )
    THETA0 ( K0 ) = TINID + FLOAT ( K0-1 )*DTHET
    THETA0 ( K0 ) = RAD*THETA0 ( K0 )
 END D0

 DW = 1.0/FLOAT ( NW-1 )
 IF ( TASKID == MASTER ) THEN
    DO I = 1, NW
       WNUM1 ( I ) = WNMIN + ( WNMAX - WNMIN )*DW*FLOAT ( I-1 )
    END DO
 END IF

 CALL MPI_SCATTER ( WNUM1, CHUNKSIZE, MPI_REAL, WNUM, CHUNKSIZE,
&                   MPI_REAL, 0, MPI_COMM_WORLD, IERR )

 DO ID = 1, CHUNKSIZE
    OMEGA_P = 45.65781
    GAMMA = 0.091169022
    OMEGA = WNUM ( ID )
    EPSR = 1.0 - OMEGA_P**2/( OMEGA**2 + GAMMA**2 )
    EPSI = ( GAMMA*OMEGA_P**2 )/( OMEGA**3 + OMEGA*GAMMA**2 )
    EPS = CMPLX ( EPSR, EPSI )
    NR = REAL ( SQRT ( EPS ) )
    NI = AIMAG ( SQRT ( EPS ) )
    NC = CMPLX ( NR, NI )

    CALL NORTE ( NORFAC, LAMBD, G, THETA0, NANG, FD0, LARGE,
   &               WNUM ( ID ) )
    CALL PERFILES ( SF, SG, DSF, DSG, DDSF, DDSG, NWA, NWB, NPT,
   &               NPA, NPB, HSLITW, HLARGE, DELTA, A1, A2, PEROPD,
   &               PHASE, PI, R, NPCILI, NPER )
    CALL KFUNC ( CIMP, DSF, DDSF, DSG, DDSG, NPT, NC, WNUM (ID) )
    CALL EINUMNR ( EI, NPT, NANG, SF, SG,THETA0, LAMBD, G, D, FDO,
   &               WNUM ( ID ) )
    CALL MATE ( ME, SF, DSF, DDSF, SG, DSG, DDSG, CIMP, NC,
   &               WNUM ( ID ), DELTA, NPT, POL )
    CALL CGESV ( NPT, NANG, ME , NPT, IWA, EI, NPTM IER )
```

```
      CALL AKAN ( AK, NFF, NANG, LARGE, THETA0, LAMBD, G, D, FDO,
     &              WNUM ( ID ) )
      CALL ENERGY ( NPT, NANG, FDO, POL, G, THETA0, LAMBD, AK, NC,
     &              CIMP, LARGE, SF, SG, DSF, DSG, DELTA, EI,
     &              NORFAC, CONSUP, ENERBAL, DTHET, REFLPOW, TANSPO,
     &              NFF, WNUM ( ID ) )

      FRECUENCY ( ID ) = WNUM ( ID )
      REFLECT ( ID ) = REFLPOW
      TRANS ( ID ) = TRANSPOW
      ENERG ( ID ) = ENERBAL
   END DO

   CALL MPI_GATHER ( FRECUENCY, CHUNKSIZE, MPI_REAL, SALIDA1,
  &           CHUNKSIZE, MPI_REAL, 0, MPI_COMM_WORLD, IERR )
   CALL MPI_GATHER ( REFLECT, CHUNKSIZE, MPI_REAL, SALIDA2,
  &           CHUNKSIZE, MPI_REAL, 0, MPI_COMM_WORLD, IERR )
   CALL MPI_GATHER ( TRANS, CHUNKSIZE, MPI_REAL, SALIDA3,
  &           CHUNKSIZE, MPI_REAL, 0, MPI_COMM_WORLD, IERR )
   CALL MPI_GATHER ( ENERG, CHUNKSIZE, MPI_REAL, SALIDA4,
  &           CHUNKSIZE, MPI_REAL, 0, MPI_COMM_WORLD, IERR

   IF ( TASKID == MASTER ) THEN
      DO I = 1, NW
         WRITE ( *,* ) SALIDA(I), SALIDA2 (I), SALIDA3 (I),SALIDA4(I)
      END DI
   END IF
   CALL MPI_FINALIZE(IERR)
END
```

IV.1.2. Algoritmo ScaLAPACK

En esta sección se muestra la implementación de un programa en FORTRAN que utiliza la biblioteca de ScaLAPACK aplicando el mismo IEM en forma paralela con la CPU. En el programa 2 se muestra el algoritmo que se implementó. Para conocer el funcionamiento del programa 2 revisar la Ref. (Lozano Trejo, 2017, 49–55). Por tal motivo, no se dará una explicación detalla del algoritmo.

```
**********************************************************************
```

Programa 2. Método Integral para una PCW de conductor real usando ScaLAPACK

```
**********************************************************************

PROGRAM PCW_REAL_MPI_SCALAPACK
   IMPLICIT NONE

----------------------------------------------------------------------
```

SECCIÓN 1.- Inicialización de variables, arreglos y declaración de rutinas externas

```
----------------------------------------------------------------------

   INTEGER NPT, NANG, NPA, NPB, NFF, NW, NPCILI, NPER, POL, FDO
   INTEGER DLEN_, IA, JA, IB, JB, M, N, MB, NB,RSRC, CSRC,
  &   MXLLDA, MXLLDB, NRHS, NBRHS, MXLOCR, MXLOCC, MXRHSC,re,co

   REAL PI, TINID, DTHET, DELTA, NWA, NWB, LAMBD, B, PHASE
   REAL PERIOD, F, R, P, WNMIN, WNMAX, NR, NI
   PARAMETER ( PI = 3.14159265358979323846264338327 95
   PARAMETER ( B = PI, PHASE = 0.5*PI, PERIOD = 2.0*PI )
   PARAMETER ( LAMBD = 1.033, NFF = 181, NWA = 28.0, NPER = 11 )
   PARAMETER ( NWB = NPER*PERIOD, DELTA = LAMBD/20.0, NANG = 1 )
   PARAMETER ( NPA = NINT ( NWA/DELTA ), F = 0.005 )
   PARAMETER ( R = SQRT ( F*PERIOD*B/PI ), P = 2.0*PI*R )
   PARAMETER ( NPCILI = INT ( P/DELTA ), NPB = NINT ( NWB/DELTA ) )
   PARAMETER ( NPT = 2*( 2*NPA + NPB ) + NPER*NPCILI )
   PARAMETER ( WNMIN = 1.0, WNMAX = 3.0, NW = 200, POL = 2, FDO=2 )

   PARAMETER (DLEN_= 9, IA = 1,JA = 1, IB = 1, JB = 1, RE = 2, CO =2,
  &   M = NPT,N = M, MB = 64, NB = MB, RSRC = 0, CSRC = 0, &
  &   MXLLDA = ( N/RE ) + ( MB-1 ), MXLLDB = ( N/RE ) + ( MB-1 ),
  &   NRHS = NANG, NBRHS = NRHS, MXRHSC = NRHS,
  &   MXLOCR = ( N/RE ) + ( MB-1 ), MXLOCC = ( N/CO ) + ( NB-1 ) )

   INTEGER ID, K0, I, IWA ( NPT ), IER
   REAL RAD, WNUM, G, FK0, DW, EPSR, EPSI, THETA0 ( NANG )
   REAL NORFAC ( NANG ), SLITW, HSLITW, HLARGE, LAERGE, DE, A1, A2
   REAL SF ( NPT ), SG ( NPT ), DSF ( NPT ), DSG ( NPT )
   REAL DDSF ( NPT ), DDSG ( NPT )
   REAL CONSUP, CONSDOWN, ENERBAL, REFLEPOW, TRANSPOW
   REAL VC, OMEGA_P, GAMMA, OMEGA
```

```
   COMPLEX NC, EPS, EI ( NPT, NANG ), AK ( NFF, NANG ), CIMP ( NPT )
   REAL JIL ( NPT ), JI2 ( NPT ), ETA1 ( NPT ), ETA2 ( NPT )

   INTEGER ICTXT, INFO, MYCOL, MYROW, NPCOL, NPROW
   INTEGER DESCA( DLEN_ ), DESCB( DLEN_ ), IPIV( MXLOCR + NB ),
  &   TC ( CO, MXLOCC + 1 ), TR ( re, MXLOCR + 1 )
   COMPLEX A( MXLLDA, MXLOCC ), BB( MXLLDB, MXRHSC )
   EXTERNAL BLACS_EXIT, BLACS_GRIDEXIT, BLACS_GRIDINFO,
  &   DESCINIT, MATINIT, PCGESV, SL_INIT, DISTRIBUCIONCICLICA,
  &   ORDENA, NORTE, PERFILES, EINUMNR, MATE, KFUNC, AKAN, ENERGY
   DATA NPROW / RE / , NPCOL / CO /

   CALL SL_INIT( ICTXT, NPROW, NPCOL )
   CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
   CALL DISTRIBUCIONCICLICA(TC,NPCOL,N,NB)
   CALL DISTRIBUCIONCICLICA(TR,NPROW,N,NB)

   NC=CMPLX ( NR, NI )
   EPSR = NR**2-NI**2
   EPSI = 2.0*NR*NI
   EPS = CMPLEX ( EPSR, EPSI )
```

SECCIÓN 2.- Parámetros generales y parámetros de ángulo de incidencia

---

```
   IF ( ( MYROW.EQ.0 ) .AND. ( MYCOL.EQ.0 ) ) THEN
      PRINT*, 'NPT = ',NPT
   END IF

   TINID = 0.0
   RAD = PI/180.0
   D = NWB
   SLITW = B
   HSLITW = 0.5*B
   HLARGE = NWA + HSLITW
   LARGE = 2.0*HLARGE
   A1 = 0.0*b
   A2 = A1
   DTHET = 1.0
   G = LARGE/5.0

   DO KO = 1, NANG
```

```
      FK0 = FLOAT ( K0 )
      THETA0 ( k0 ) = TINID + FLOAT ( K0 - 1 )*DTHET
      THETA0 ( K0 ) = RAD*THETA0 ( K0 )
   END DO

   DW = 1.0/ FLOAT ( NW - 1 )
```

SECCIÓN 3.- Ciclo de variación de frecuencias y llamado a las rutinas encargadas de dar vida a la simulación

---

```
   DO ID = 1, NW
      WNUM = WNMIN + ( WNMAX - WNMIN )*DW*FLOAT( ID -1 )
      OMEGA_P = 45.65781
      GAMMA = 0.091169022
      OMEGA = WNUM
      EPSR = 1.0 - OMEGA_P**2/( OMEGA**2 + GAMMA**2 )
      EPSI = ( GAMMA*OMEGA_P**2 )/( OMEGA**3 + OMEGA*GAMMA**2 )
      EPS = CMPLX ( EPSR, EPSI )
      NR = REAL ( SQRT ( EPS ) )
      NI = AIMAG ( SQRT ( EPS ) )
      NC = CMPLX ( NR, NI )

      CALL NORTE ( NORFAC, LAMBD, G, THETA0, NANG, FD0, LARGE, WNUM )
      CALL PERFILES ( SF, SG, DSF, DSG, DDSF, DDSG, NWA, NWB, NPT,
  &      NPA, NPB, HSLITW, HLARGE, DELTA, A1, A2, PERIOD, PHASE, PI,
  &      R, NPCILI, NPER )
      CALL KFUNC ( CIMP, DSF, DDSF, DSG, DDSG, NPT, NC, WNUM )
      CALL AKAN ( AK, NFF, NANG, LARGE, THETA0, LAMBD, G, D, FD0,
  &      WNUM )

      CALL DESCINIT( DESCA, M, N, MB, NB, RSRC, CSRC, ICTXT, MXLLDA,&
         INFO )
      CALL DESCINIT( DESCB, N, NRHS, MB, NRHS, RSRC, CSRC, ICTXT,&
         MXLLDB, INFO )
      CALL MATINIT( A, DESCA, BB, DESCB, N, NPCOL, NPROW, TC, TR,&
         NPT, NANG, SF, SG, WNUM, G, D, DSF, DSG, DDSF, DDSG,&
         DELTA, THETA0, LAMBD, FD0, CIMP, NC, POL )

      CALL PCGESV( N, NRHS, A, IA, JA, DESCA, IPIV, BB, IB, JB,&
         DESCB,INFO )
```

```
------------------------------------------------------------------------
```

SECCIÓN 4.- Parte encargada de organizar la solución

```
------------------------------------------------------------------------

      IF ( ( MYROW.EQ.0 ) .AND. ( MYCOL.EQ.0 ) ) THEN
         DO I = 0, NPROW-1
            IF ( I .EQ. 0 )THEN
               CALL ORDENA(EI,BB,I,0,N,NRHS,MXLLDB,MXRHSC,TR,NPROW)
            ELSE
               CALL CGERV2D(ICTXT, MXLLDB, MXRHSC, BB, MXLLDB, I, 0 )
               CALL ORDENA(EI,BB,I,0,N,NRHS,MXLLDB,MXRHSC,TR,NPROW)
            END IF
         END DO
      END IF
      IF (MYROW .NE. 0 .AND. MYCOL .EQ. 0 )THEN
         CALL CGESD2D( ICTXT, MXLLDB, MXRHSC, BB, MXLLDB, 0, 0 )
      END IF


------------------------------------------------------------------------
```

SECCIÓN 5.- Cálculo de la potencia media reflejada, transmitida y la conservación de la energía de la PCW

```
------------------------------------------------------------------------

      IF ( ( MYROW.EQ.0 ) .AND. ( MYCOL.EQ.0 ) ) THEN
         CALL ENERGY ( NPT, NANG, FDO, POL, G, THETA0, LAMBD, AK,&
            NC, CIMP, LARGE, SF, SG, DSF, DSG, DELTA, EI, NORFAC,&
            CONSUP, CONSDOWN, ENERBAL, DTHET, REFLPOW, TRANSPOW,&
            NFF, WNUM )
         PRINT*,id,'FREQUENCY =', WNUM
         PRINT*,'MEAN REFLECTED POWER =', REFLPW
         PRINT*,'MEAN TRANSMITTED POWER =', TRANSPOW
         PRINT*,'MEAN SCATTERED POWER =',ENERBAL
      END IF
   END DO
   CALL BLACS_EXIT( 0 )
END
```

Los algoritmos más importantes del programa pueden ser consultadas en el apéndice A.

IV.1.3. Algoritmo CUDA

En esta sección se muestra la implementación de un programa en CUDA C aplicado al mismo Método Integral en forma paralela. En el programa 3 se muestra el algoritmo que se implementó para calcular la respuesta óptica de la PCW de conductor real utilizando la GPU.

****

Programa 3. Método Integral para una PCW de conductor real usando CUDA C

****

```
-------------------------------------------------------------------------
```

SECCIÓN 1.- Declaración de librerías externas.

```
-------------------------------------------------------------------------

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "subrutinas.cu"
#include <time.h>
#include <sys/time.h>
#include <cuda.h>
#include <cusolverDn.h>
#include <cuComplex.h>
#include <assert.h>

int main(){

-------------------------------------------------------------------------
```

SECCIÓN 2.- Declaración y inicialización de variables.

```
-------------------------------------------------------------------------

   int npt,nang,npa,npb,nff,nw,npcili,nper,pol,fdo;
   float pi,tinid,dthet,delta,nwa,nwb,b,phase,period,f,r,p;
   float lambd,wnmin,wnmax;

   cusolverDnHandle_t cusolverH = NULL;
```

```
cudaStream_t stream = NULL;
cusolverStatus_t status = CUSOLVER_STATUS_SUCCESS;
cudaError_t cudaStat1 = cudaSuccess;
cudaError_t cudaStat3 = cudaSuccess;
cudaError_t cudaStat4 = cudaSuccess;
int *d_Ipiv = NULL; /* pivoting sequence */
int *d_info = NULL; /* error info */
int lwork = 0;     /* size of workspace */
cuComplex *d_work = NULL; /* device workspace for getrf */
const int pivot_on = 0;

pi=M_PI;
b=1.0*pi;
phase=0.5*pi;
period=2.0*pi;
lambd=1.033;//Metal
nff=181;
nwa=28.0;
nper=5;//*****
nwb=nper*period;
delta =lambd/20.0;
nang=1;
npa=(int)nwa/delta;
f=0.005;
r=sqrt(f*period*b/pi);
p=2.0*pi*r;
npcili=(int)(p/delta);
npb=(int)(nwb/delta);
npt=2*(2*npa+npb)+nper*npcili;

wnmin=1.0f;
wnmax=3.0f;
nw=200;
pol=2;
fdo=2;

int k0,id;
float rad,wnum,g,dw,epsr,epsi,hslitw,hlarge,large;
float d,A1,A2,omega_p,gamma,omega;
float theta0[nang],norfac[nang],sf[npt],sg[npt],dsf[npt],dsg[npt];
float ddsf[npt],ddsg[npt],variables[3];
cuComplex nc,eps;

cuComplex* me_d;
cuComplex* ei_d;
```

```
cuComplex* cimp_d;
cuComplex* ak_h;
cuComplex* ei_h;
cuComplex* cimp_h;
float* sf_d;
float* sg_d;
float* dsf_d;
float* dsg_d;
float* ddsf_d;
float* ddsg_d;
float* theta0_d;

ak_h=(cuComplex *)malloc(nff*nang*sizeof(cuComplex));
ei_h=(cuComplex *)malloc(npt*nang*sizeof(cuComplex));
cimp_h=(cuComplex *)malloc(npt*sizeof(cuComplex));
```

SECCIÓN 3.- Asignación de memoria de los arreglos que se encontrarán almacenados en la GPU.

```
cudaMalloc((void**)&me_d,npt*npt*sizeof(cuComplex));
cudaMalloc((void**)&ei_d,npt*nang*sizeof(cuComplex));
cudaMalloc((void**)&cimp_d,npt*sizeof(cuComplex));
cudaMalloc((void**)&theta0_d,nang*sizeof(float));
cudaMalloc((void**)&sf_d,npt*sizeof(float));
cudaMalloc((void**)&sg_d,npt*sizeof(float));
cudaMalloc((void**)&dsf_d,npt*sizeof(float));
cudaMalloc((void**)&dsg_d,npt*sizeof(float));
cudaMalloc((void**)&ddsf_d,npt*sizeof(float));
cudaMalloc((void**)&ddsg_d,npt*sizeof(float));

cudaStat3 = cudaMalloc ((void**)&d_Ipiv, sizeof(int) * npt);
cudaStat4 = cudaMalloc ((void**)&d_info, sizeof(int));
assert(cudaSuccess == cudaStat3);
assert(cudaSuccess == cudaStat4);
```

SECCIÓN 4.- Parámetros generales.

```
printf("radio= %f  npcili=%d  npt=%d\n",r,npcili,npt);
rad=pi/180.0;
d=nwb;
hslitw=0.5*b;
hlarge=nwa+hslitw;
large=2.0*hlarge;
A1=0.0*b;
A2=0.0*b;

g=large/5.0;
dthet=1.0;
tinid=0.0;
```

SECCIÓN 5.- Llenado del arreglo que almacena los ángulos a los cuales se hará incidir el haz.

```
for (k0=1; k0<=nang; k0++) {
   theta0[k0-1]=tinid+(k0-1.0)*dthet;
   theta0[k0-1]=rad*theta0[k0-1];
}
```

SECCIÓN 6.- Ciclo principal encargado de realizar las variaciones de frecuencia, llamado a las rutinas y copia de arreglos de la memoria de la máquina a la memoria de la GPU.

```
dw=1.0/(nw-1.0);
for (id=1; id<=nw; id++) {
   wnum=wnmin+(wnmax-wnmin)*dw*(id-1.0);
   omega_p=45.65781;//reduced plasma frequency for silver
   gamma=0.091169022;//reduced damping frequency for silver
   omega=wnum;
   epsr=1.0 - (omega_p*omega_p)/(omega*omega+gamma*gamma);
   epsi=(gamma*omega_p*omega_p)/(omega*omega*omega+omega*
      gamma*gamma);
   eps.x=epsr;
   eps.y=epsi;
   nc=cuCsqrt(eps);
```

```
norte(norfac,lambd,g,theta0,nang,fdo,large,wnum);
perfiles(sf,sg,dsf,dsg,ddsf,ddsg,nwa,nwb,npt,npa,npb,
   hslitw,hlarge,delta, A1,A2,period,phase,pi,r,npcili,nper);
akan(ak_h,nff,nang,large,theta0,lambd,g,d,fdo,wnum);

kfunc(cimp_h,dsf,ddsf,dsg,ddsg,npt,nc,wnum);
cudaMemcpy(cimp_d,cimp_h,npt*sizeof(cuComplex),
   cudaMemcpyHostToDevice);
cudaMemcpy(sf_d,sf,npt*sizeof(float),
   cudaMemcpyHostToDevice);
cudaMemcpy(sg_d,sg,npt*sizeof(float),
   cudaMemcpyHostToDevice);
cudaMemcpy(dsf_d,dsf,npt*sizeof(float),
   cudaMemcpyHostToDevice);
cudaMemcpy(dsg_d,dsg,npt*sizeof(float),
   cudaMemcpyHostToDevice);
cudaMemcpy(ddsf_d,ddsf,npt*sizeof(float),
   cudaMemcpyHostToDevice);
cudaMemcpy(ddsg_d,ddsg,npt*sizeof(float),
   cudaMemcpyHostToDevice);
cudaMemcpy(theta0_d,theta0,nang*sizeof(float),
   cudaMemcpyHostToDevice);
```

SECCIÓN 7.- Definición del tamaño de la malla y de los bloques que se utilizarán, llamado a las rutinas encargadas de llenar la matriz de ecuaciones.

```
dim3 blockSize (32,32,1);
dim3 gridSize (floor(npt/32)+32,floor(npt/32)+32,1);
mateYeinumnr_kernel<<<gridSize,blockSize>>>
(me_d,sf_d,dsf_d,ddsf_d,sg_d,dsg_d,ddsg_d,cimp_d,
wnum,delta,npt,pol,ei_d,nang,theta0_d,g,d);
```

SECCIÓN 7.1.- Encargada de calcular la solución al sistema.

```
/* create cusolver handle, bind a stream */
```

```
      status = cusolverDnCreate(&cusolverH);
      assert(CUSOLVER_STATUS_SUCCESS == status);

      cudaStat1 = cudaStreamCreateWithFlags(&stream,
         cudaStreamNonBlocking);
      assert(cudaSuccess == cudaStat1);

      status = cusolverDnSetStream(cusolverH, stream);
      assert(CUSOLVER_STATUS_SUCCESS == status);

/* query working space of getrf */
      status = cusolverDnCgetrf_bufferSize(cusolverH,npt,npt,me_d,
         npt,&lwork);
      assert(CUSOLVER_STATUS_SUCCESS == status);

      cudaStat1 = cudaMalloc((void**)&d_work, sizeof(cuComplex)*
         lwork);
      assert(cudaSuccess == cudaStat1);

/* LU factorization */
      if (pivot_on){
         status = cusolverDnCgetrf(cusolverH,npt,npt,me_d,npt,d_work,
            d_Ipiv,d_info);
      }
      else{
         status = cusolverDnCgetrf(cusolverH,npt,npt,me_d,npt,d_work,
            NULL,d_info);
      }
      cudaStat1 = cudaDeviceSynchronize();
      assert(CUSOLVER_STATUS_SUCCESS == status);
      assert(cudaSuccess == cudaStat1);
/* solve A*X = B */
      if (pivot_on){
         status = cusolverDnCgetrs(cusolverH,CUBLAS_OP_N,npt,nang,
            /* nrhs */me_d,npt,d_Ipiv,ei_d,npt,d_info);
      }
      else{
         status = cusolverDnCgetrs(cusolverH,CUBLAS_OP_N,npt,nang,
            /* nrhs */me_d,npt,NULL,ei_d,npt,d_info);
      }
      cudaStat1 = cudaDeviceSynchronize();
      assert(CUSOLVER_STATUS_SUCCESS == status);
      assert(cudaSuccess == cudaStat1);

      cudaMemcpy(ei_h,ei_d,npt*nang*sizeof\qquad
```

```
      (cuComplex),cudaMemcpyDeviceToHost);
```

SECCIÓN 8.- En esta sección se calcula la potencia media reflejada y transmitida para verificar la conservación de la energía y por último, pero no menos importante la liberación de la memoria de la GPU.

```
      energy(npt,nang,fdo,pol,g,theta0,lambd,ak_h,nc,cimp_h,
      large,sf,sg,dsf,dsg,delta,ei_h,norfac,dthet,nff,
      variables,wnum);
      printf("id: %d      Frequency= %f\n",id,wnum);
      printf("MEAN REFLECTED POWER = %.10f\nMEAN TRANSMITTED POWER =
         %.7e\nMEAN SCATTERED POWER = %.10f\n",variables[0],
         variables[1],variables[2]);
   }
   cudaFree(me_d);
   cudaFree(cimp_d);
   cudaFree(sf_d);
   cudaFree(sg_d);
   cudaFree(dsf_d);
   cudaFree(dsg_d);
   cudaFree(ddsf_d);
   cudaFree(ddsg_d);
   cudaFree(ei_d);
   cudaFree(theta0_d);
   cudaFree(ak_h);
   cudaFree(ei_h);
   cudaFree(cimp_h);
   return 0;
}
```

En general este programa es muy similar al descrito en la Ref. (Lozano Trejo, 2017, 55–61); por lo tanto nada más nos enfocaremos en las diferencias principales, las cuales se encuentran en la sección 7.1 del código. En esta subsección se inicializan las subrutinas necesarias para el funcionamiento de las rutinas *cusolverDnCgetrf* y *cusolverDnCgetrs*, donde la primera es la encargada de realizar la factorización LU y la segunda se encarga de resolver el sistema utilizando la factorización LU antes calculada. Por último se

encuentra la rutina encargada de copiar la solución del sistema de la memoria de la GPU a la memoria de la CPU. Los algoritmos más importantes del programa pueden ser consultadas en el apéndice B.

IV.1.4. Algoritmo ScaLAPACK y CUDA

En esta sección se muestra la implementación de un programa que combina ScaLAPACK con CUDA aplicado al mismo Método Integral. En el programa 4 se muestra el algoritmo que se implementó para calcular la respuesta óptica de la PCW de conductor real combinando el hardware de la CPU y la GPU.

****

Programa 4. Método Integral para una PCW de conductor real usando SCALAPACK y CUDA

****

```
------------------------------------------------------------------------
```

SECCIÓN 1.- Declaración de librerías externas.

```
------------------------------------------------------------------------

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <complex.h>
#include "mpi.h"
#include "subrutina.c"
#include "externas.c"
#include <mkl_scalapack.h>

main(int argc, char** argv) {

------------------------------------------------------------------------
```

SECCIÓN 2.- Declaración y inicialización de variables.

```
------------------------------------------------------------------------

   MPI_Status estado;
```

```
int npt,nang,npa,npb,nff,nw,npcili,nper,op,g;
float pi,tinid,dthet,delta,nwa,nwb,b,phase,period,f,r,p;
float lambd,wnmin,wnmax;

int DLEN,IA,JA,IB,JB,M,N,MB,NB,RSRC,CSRC,MXAM,MXAN,MXBM;
int NRHS,NBRHS,NPROC_ROWS,NPROC_COLS,MY_RANK,MXLOCR,MXLOCC;
int MY_PROCESS_ROW,MY_PROCESS_COL,ICTXT,NPROCS_MPI,INFO;
int i;
int j;

pi=M_PI;
b=1.0*pi;
phase=0.5*pi;
period=2.0*pi;
lambd=1.033;
nff=181;
nwa=28.0;
nper=10;
nwb=nper*period;
delta =lambd/20.0;
nang=1;
npa=(int)nwa/delta;
f=0.005;
r=sqrt(f*period*b/pi);
p=2.0*pi*r;
npcili=(int)(p/delta);
npb=(int)(nwb/delta);
npt=2*(2*npa+npb)+nper*npcili;

DLEN=9,IA=1,JA=1,IB=1,JB=1,RSRC=0,CSRC=0;
M=npt,N=M,MB=64,NB=64,NRHS=nang;
NPROC_ROWS=2,NPROC_COLS=2;

int k0,id,pol,fdo,ier;
float rad,wnum,fk0,hslitw,slitw,hlarge,large,dw,d,A1,A2;
float consup,consdown;
float theta0[nang],norfac[nang],sf[npt],sg[npt],dsf[npt];
float dsg[npt],ddsf[npt],ddsg[npt],variables[3];
float complex iwa[npt];
float epsr,epsi,omega_p,gamma,omega,nr,ni;
float complex nc,eps;

float complex* A_LOCAL;
float complex* B_LOCAL;
float complex* ak;
```

```
int* IPIV;
int* DESCA;
int* DESCB;
int* TR;
int* TC;
int* TEM;
int* TR_D;
int* TC_D;
int* DESCA_D;
int* DESCB_D;

float complex* A_LOCAL_D;
float complex* B_LOCAL_D;
float complex* ei_h;
float complex* cimp_h;
float complex* cimp_d;
float* sf_d;
float* sg_d;
float* dsf_d;
float* dsg_d;
float* ddsf_d;
float* ddsg_d;
float* theta0_d;
float complex* v;
```

SECCIÓN 3.- Inicialización de MPI y ScaLAPACK

```
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &NPROCS_MPI);
MPI_Comm_rank(MPI_COMM_WORLD, &MY_RANK);

Cblacs_get(0, 0, &ICTXT);
Cblacs_gridinit(&ICTXT, "R", NPROC_ROWS, NPROC_COLS);
Cblacs_pcoord(ICTXT, MY_RANK, &MY_PROCESS_ROW,&MY_PROCESS_COL);

MXLOCR = (N/NPROC_ROWS)+(MB);
MXLOCC = (N/NPROC_COLS)+(NB);
MXAM = numroc_(&M, &MB, &MY_PROCESS_ROW, &RSRC, &NPROC_ROWS);
MXAN = numroc_(&N, &NB, &MY_PROCESS_COL, &CSRC, &NPROC_COLS);
NBRHS = numroc_(&NRHS, &NB, &MY_PROCESS_COL, &CSRC, &NPROC_COLS);
MXBM = numroc_(&M, &MB, &MY_PROCESS_ROW, &RSRC, &NPROC_ROWS);
```

SECCIÓN 4.- Asignación de memoria de los arreglos que se encontrarán almacenados en la GPU y CPU.

```
   cudaMalloc((void**)&cimp_d,npt*sizeof(float complex));
   cudaMalloc((void**)&theta0_d,nang*sizeof(float));
   cudaMalloc((void**)&sf_d,npt*sizeof(float));
   cudaMalloc((void**)&sg_d,npt*sizeof(float));
   cudaMalloc((void**)&dsf_d,npt*sizeof(float));
   cudaMalloc((void**)&dsg_d,npt*sizeof(float));
   cudaMalloc((void**)&ddsf_d,npt*sizeof(float));
   cudaMalloc((void**)&ddsg_d,npt*sizeof(float));

   cudaMalloc((void**)&TR_D,NPROC_ROWS*MXLOCR*sizeof(int));
   cudaMalloc((void**)&TC_D,NPROC_COLS*MXLOCC*sizeof(int));
   cudaMalloc((void**)&DESCA_D,9*sizeof(int));
   cudaMalloc((void**)&DESCB_D,9*sizeof(int));
   cudaMalloc((void**)&A_LOCAL_D,MXAM*MXAN*sizeof(float complex));
   cudaMalloc((void**)&B_LOCAL_D,MXBM*NBRHS*sizeof(float complex));

   ei_h=(float complex *)malloc(npt*nang*sizeof(float complex));
   ak=(float complex *)calloc(nff*nang,sizeof(float complex));
   cimp_h=(float complex *)malloc(npt*sizeof(float complex));
   A_LOCAL = (float complex *)calloc(MXAM*MXAN,sizeof(float complex));
   B_LOCAL = (float complex *)calloc(MXBM*NBRHS,sizeof(float complex));
   IPIV = (int *)malloc((MXAM + NB)*sizeof(int));
   DESCA=(int *)malloc(DLEN*sizeof(int));
   DESCB=(int *)malloc(DLEN*sizeof(int));
   TR=(int *) calloc(NPROC_ROWS*MXLOCR,sizeof(int));
   TC=(int *) calloc(NPROC_COLS*MXLOCC,sizeof(int));
   TEM=(int *) calloc(NPROC_ROWS,sizeof(int));
   v=(float complex*)malloc(5*sizeof(float complex));
   for(i=0;i<NPROC_ROWS;i++){
      TEM[i]=numroc_(&M, &MB, &i, &RSRC, &NPROC_ROWS);
   }
```

SECCIÓN 5.- Inicialización de la distribución cíclica, parámetros generales y de ángulo de incidencia.

```
DISTRIBUCIONCICLICA(TC,NPROC_COLS,N,NB);
DISTRIBUCIONCICLICA(TR,NPROC_ROWS,M,MB);

wnmin=1.0;
wnmax=3.0;
nw=200;
op=4;
pol=2;
fdo=2;

if(MY_PROCESS_ROW==0 && MY_PROCESS_COL==0){
   printf("radio= %f  npcili=%d  npt=%d\n",r,npcili,npt);
}

rad=pi/180.0;
d=nwb;
slitw=b;
hslitw=0.5*b;
hlarge=nwa+hslitw;
large=2.0*hlarge;
A1=0.0*b;
A2=0.0*b;
wnum=wnmin;
g=2.0*nwa/5.0;

dthet=1.0;
tinid=0.0;
for (k0=1; k0<=nang; k0++) {
   fk0=k0*1.0;
   theta0[k0-1]=tinid+(k0-1.0)*dthet;
   theta0[k0-1]=rad*theta0[k0-1];
}
dw=1.0/(nw-1.0);
```

SECCIÓN 6.- Inicialización del ciclo que varia las frecuencias y llamado a las rutinas encargadas de dar vida a la simulación.

```
for (id=1; id<=nw; id++) {
   wnum=wnmin+(wnmax-wnmin)*dw*(id-1.0);
   omega_p=45.65781;//reduced plasma frequency for silver
```

```
gamma=0.091169022;//reduced damping frequency for silver
omega=wnum;
epsr=1.0 - (omega_p*omega_p)/(omega*omega+gamma*gamma);
epsi=(gamma*omega_p*omega_p)/(omega*omega*omega+omega*gamma*gamma);
eps=epsr+epsi*I;
nc=csqrtf(eps);
nr=creal(nc);
ni=cimag(nc);
norte(norfac,lambd,g,theta0,nang,fdo,large,wnum);
perfiles(sf,sg,dsf,dsg,ddsf,ddsg,nwa,nwb,npt,
npa,npb,hslitw,hlarge,delta, A1,A2,period,phase,pi,r,
npcili,nper);
akan(ak,nff,nang,large,theta0,lambd,g,d,fdo,wnum);
kfunc(cimp_h,dsf,ddsf,dsg,ddsg,npt,nc,wnum);

v[0]=delta;
v[1]=wnum;
v[2]=g;
v[3]=d;
v[4]=nc;

descinit_( DESCA, &M, &N, &MB, &NB, &RSRC, &CSRC, &ICTXT,
  &MXAM, &INFO);
descinit_( DESCB, &M, &NRHS, &MB, &NRHS, &RSRC, &CSRC,
  &ICTXT, &MXBM, &INFO);
MATINIT_CUDA(A_LOCAL,B_LOCAL,A_LOCAL_D,B_LOCAL_D,cimp_h,cimp_d,v,
  sf,sf_d,dsf,dsf_d,ddsf,ddsf_d,sg,sg_d,dsg,dsg_d,ddsg,ddsg_d,
  theta0,theta0_d,DESCA,DESCA_D,DESCB,DESCB_D,TR,TR_D,TC,TC_D,
  npt,nang,fdo,pol,NPROC_ROWS,NPROC_COLS,NBRHS,MXLOCR,MXLOCC,
  MXAM,MXAN,MXBM,MY_PROCESS_ROW,MY_PROCESS_COL);
pcgesv_(&N, &NRHS, A_LOCAL, &IA, &JA, DESCA, IPIV, B_LOCAL, &IB,
  &JB, DESCB, &INFO);
MPI_Barrier(MPI_COMM_WORLD);
```

SECCIÓN 7.- Parte encargada de organizar la solución.

```
if(MY_PROCESS_ROW==0 && MY_PROCESS_COL==0){
   for(i=0;i<NPROC_ROWS;i++){
      if(i==0){
         ORDENA(ei_h,B_LOCAL,i,MY_PROCESS_COL,M,NRHS,MXBM,
           NBRHS,TR,TC,NPROC_ROWS,NPROC_COLS);
```

```
            }
            else{
               MPI_Recv(B_LOCAL,TEM[i]*NBRHS,MPI_C_COMPLEX,
                 i*NPROC_ROWS,1,MPI_COMM_WORLD,&estado);
               ORDENA(ei_h,B_LOCAL,i,MY_PROCESS_COL,M,NRHS,TEM[i],
                 NBRHS,TR,TC,NPROC_ROWS,NPROC_COLS);
            }
         }
      }
      if(MY_PROCESS_ROW!=0 && MY_PROCESS_COL==0){
         MPI_Send(B_LOCAL,MXBM*NBRHS,MPI_C_COMPLEX,0,1,
         MPI_COMM_WORLD);
      }
      MPI_Barrier(MPI_COMM_WORLD);
```

SECCIÓN 8.- Cálculo de la potencia media reflejada, transmitida y la conservación de la energía.

```
      if(MY_PROCESS_ROW==0 && MY_PROCESS_COL==0){
         energy(npt,nang,fdo,pol,g,theta0,lambd,ak,nc,cimp_h,
           large,sf,sg,dsf,dsg,delta,ei_h,norfac,dthet,
           nff,variables,wnum);
         printf("id: %d      Frequency= %f\n",id,wnum);
         printf("MEAN REFLECTED POWER = %.10f\n
           MEAN TRANSMITTED POWER = %.7e\nMEAN SCATTERED POWER = %.10f\n",
           variables[0],variables[1],variables[2]);
      }
   }
   Cblacs_exit(1);
   cudaFree(A_LOCAL_D);
   cudaFree(sf_d);
   cudaFree(sg_d);
   cudaFree(dsf_d);
   cudaFree(dsg_d);
   cudaFree(ddsf_d);
   cudaFree(ddsg_d);
   cudaFree(B_LOCAL_D);
   cudaFree(theta0_d);
   cudaFree(cimp_d);
   MPI_Finalize();
}
```

A continuación explicaremos en más detalle cada parte del Programa 4. Algunas secciones serán omitidas en la explicación por que se encuentran explicadas en la Ref. (Lozano Trejo, 2017).

Sección 1

En esta sección se declaran las librerías externas que son necesarias para el correcto funcionamiento del programa.

Sección 2

En esta sección del programa se inicializan las variables y arreglos. Para mayor información consultar la Ref. (Lozano Trejo, 2017, 53–54).

Sección 3

En esta sección se inicializan MPI, ScaLAPACK y se calculan parámetros que serán utilizados por otras rutinas.

Sección 4

En esta sección se asigna la memoria de los arreglos que se encontrarán almacenados en la GPU y en la CPU. Además se calculan parámetros que serán utilizados por otras rutinas.

Sección 5

En esta sección se encuentra los parámetros generales que serán utilizados en el programa. También se encuentra el ciclo que hace el llenado del arreglo $theta0$, el cual contiene los ángulos a los cuales se hará incidir el haz Gaussiano a la PCW. En nuestro programa sólo utilizaremos el ángulo de incidencia normal (0°).

Sección 6

En esta sección se inicializa el ciclo principal, el cual realiza las variaciones de la frecuencia en intervalos DW. A continuación daremos una pequeña explicación de cual es la función de cada rutina.

norte.- Esta rutina se encarga de calcular el factor de normalización.

perfiles.- Esta rutina genera el perfil geométrico de la PCW calculando sus coordenadas paramétricas; así como su primera y segunda derivada.

akan.- Esta rutina calcula el espectro del campo incidente de forma analítica.

kfunc.- Esta rutina calcula la condición a la frontera de impedancia.

descinit_.- Esta rutina inicializa el descriptor de cada matriz (ver apéndice C).

MATINIT_CUDA.- Esta rutina se utiliza como puente entre la CPU y la GPU; a su vez llama a la rutina encargada de llenar las submatrices A_LOCAL y B_LOCAL utilizando la GPU (ver apéndice C).

pcgesv_.- Esta rutina es la encargada de calcular la solución del sistema de ecuaciones lineales complejas.

Sección 7

En esta sección se ordena la solución dada por la rutina pcgesv_.

Sección 8

En esta sección se calcula la potencia media reflejada y transmitida para verificar la conservación de la energía y por último, pero no menos importante la liberación de la memoria de la GPU.

IV.2. Tiempo de Cómputo

A continuación se muestran los resultados obtenidos para diferentes versiones de los algoritmos tanto en su forma secuencial como en paralelo del IEM para estudiar la respuesta óptica de una PCW de conductor real. En la tabla I se muestra los resultados del tiempo de cómputo en segundos para una matriz cuadrada variando el número de periodos de la PCW y utilizando siete distintas formas de programación. La primera es la secuencial, que es la forma tradicional de programación. Para la forma secuencial se tienen dos columnas distintas: una es usando las librerías de LAPACK y la otra es utilizando la librerías de Intel (MKL). Las otras cinco formas de programación usan el paradigma de paralelización al utilizar distintas librerías, distintos grados de paralelización y combinando distintos elementos del hardware. Primeramente se utilizó OpenMP ya que realizar la paralelización es muy sencilla y no requiere grandes cambios al código que se utiliza para la forma secuencial. Para nuestro caso sólo se utilizó al momento de construir la matriz. Para MPI se usa la comunicación colectiva que en este caso la parte que se paraleliza es el ciclo principal, el cual es el encargado de calcular la respuesta óptica para las 200 frecuencias usando el IEM. Otra de las formas de paralelización que se utilizó, fue el uso de la librería de ScaLAPACK. Para este caso se paralelizó la inversión como la construcción de la matriz. Las formas antes mencionadas se centran en utilizar el procesador o CPU para realizar todas las tareas y las siguientes formas utilizan la tarjeta gráfica o GPU tanto para construir la matriz así como para la inversión. Por último se utilizó una combinación de CPU con GPU, donde la GPU se utilizó para construir la matriz y la CPU para la inversión de matriz.

Como se esperaba, en la Fig. 8 podemos notar que los tiempos de cómputo llevados acabo por las versiones en paralelo son mucho más pequeños que las versiones en secuencial. Esto se debe a que se hace un uso más eficiente de los recursos de la

máquina.

Tabla I: Tiempo de cómputo para calcular la respuesta óptica de una PCW de conductor real variando el número de periodos usando algoritmos del método integral en las formas secuencial y paralela.

Entrada	Num. Periodos	Secuencial LAPACK (s)	Secuencial MKL (INTEL) (s)	OpenMP (s)	Scalapack (Inv. + Cons.)(s)	MPI Colectiva (s)	CUDA (Cons.) Y ScaLAPACK (Inv.)(s)	Paralelo C (Inv. + Cons.) (GPU)(s)
3489 × 3489	5	15223	309	178	625	140	95	36
4810 × 4810	10	41653	617	420	1001	323	248	60
7452 × 7452	20	147812	2660	1291	2530	1045	893	139
10096 × 10096	30	354298	5360	3236	4871	2412	2183	277
12740 × 12740	40	708645	9681	6588	6260	4584	4297	504
15382 × 15382	50	–	17357	11728	9909	7920	7541	849

De igual manera vamos hacer una comparación de todas las versiones respecto a la versión secuencial LAPACK tomando en consideración la matriz de rango 12740 × 12740. Primero, con la versión secuencial usando la librería MKL de Intel se obtuvo una rapidez de 73 veces con respecto a la versión secuencial de LAPACK. Ahora haciendo la comparación con las versiones paralelas que usan la CPU, podemos observar que la versión de OpenMP obtuvo una rapidez de 107 veces; para la versión de ScaLAPACK obtuvo una rapidez de 113 veces; para MPI con comunicación Colectiva obtuvo una rapidez de 154 veces y para la versión que utiliza la programación en paralelo en la GPU se obtuvo una rapidez de 1406 veces más rápido. Finalmente para la versión que combina el uso de la CPU con la GPU se obtuvo una rapidez de 164 veces más rápido. Esto nos muestra que la mejor forma de estudiar la respuesta óptica de una PCW de conductor real es utilizando la GPU. Por otro lado, si nada más se quiere utilizar CPU, la librería de MPI es la mejor forma. Sin embargo, tanto la GPU como los procesadores usando MPI tienen una pequeña desventaja respecto a las versiones secuenciales y la paralela usando ScaLAPACK u OpenMP, la cual es el tamaño de la memoria requerida para funcionar. Para solventar este problema y tratando de perder el menor rendimiento posible se optó por combinar las mejores cualidades de la CPU y GPU, las cuales son que la CPU suele tener mayor memoria a su disposición y la GPU tiene la posibilidad de realizar una gran cantidad de tareas al mismo tiempo. Teniendo estas cualidades en mente se decidió que la GPU se encargaría de construir las submatrices que utiliza ScaLAPACK y ScaLAPACK se utilizaría para invertir la matriz. De esta manera el proceso de construcción e inversión puede escalarse a medida que el problema crece.

En la Tabla II y en la Fig. 9 se puede observar que al aumentar el número de procesadores el tiempo de cómputo requerido disminuye. También se observa una irregularidad en las últimas dos mediciones. Esto se debe a que en la computadora donde se realizaron los cálculos cuenta con 40 hilos de procesamiento, de los cuales 20

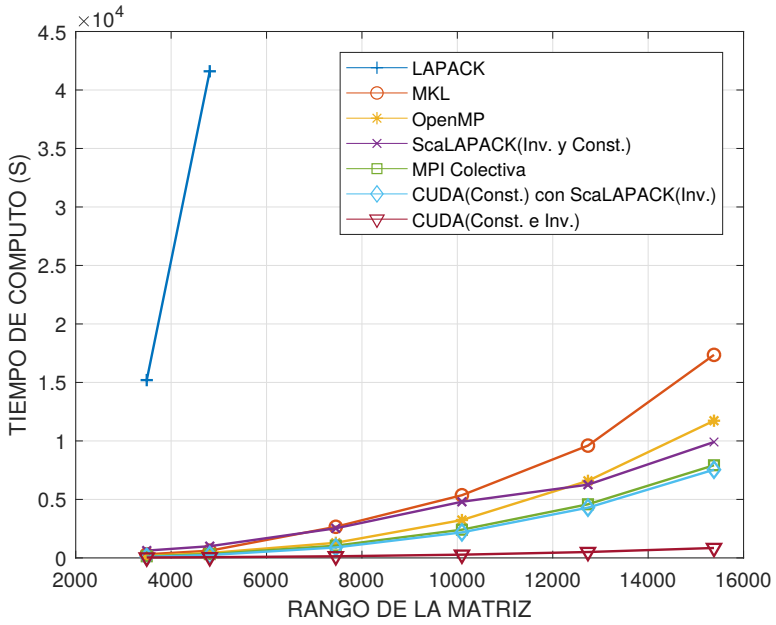

Figura 8: Tiempo de cómputo en segundos para el cálculo de la respuesta óptica en forma secuencial y paralela variando el número de períodos de la PCW de conductor real.

son físicos y los otros 20 son virtuales, lo que nos lleva a una pérdida de rendimiento.

Tabla II: Tiempo de cómputo requerido para calcular la reflectancia usando MPI variando el número de procesadores.

Número de Procesadores	**Tiempo de cómputo (s)**
2	23338
4	12156
8	6484
10	5356
20	3020
25	4417
40	3957

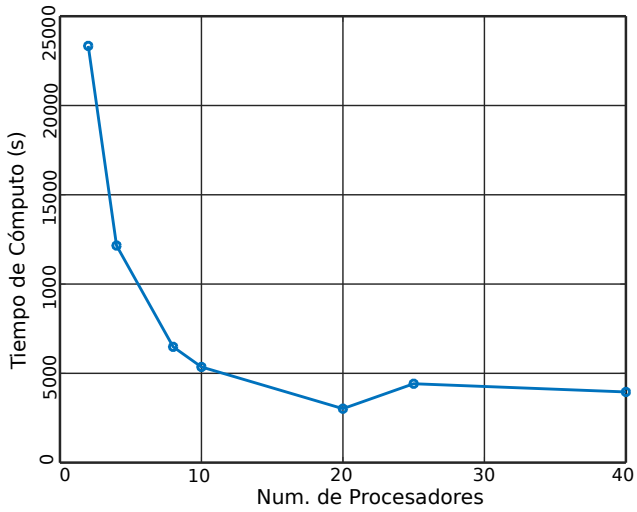

Figura 9: Tiempo de cómputo en segundos para el cálculo de la respuesta óptica variando el número de procesadores.

Para este trabajo se utilizaron dos computadoras: la primera cuenta con un procesador i7-700HQ, una tarjeta gráfica GeForce GTX 1060 con 6 GB de memoria VRAM y 16 GB de memoria RAM; la segunda cuenta con dos procesadores Xeon e5-2640 v2, una tarjeta gráfica TESLA k40 con 12 GB de memoria VRAM y 64 GB de memoria RAM.

Capítulo V

MÉTODO DE ELEMENTOS FINITOS

En este capítulo se describe un método numérico general para aproximar soluciones de ecuaciones diferenciales parciales. Este método se conoce como el Método de Elementos Finitos (FEM). En un capítulo posterior se utilizará para resolver la ecuación de Helmholtz en 1D y 2D con distintas geometrías.

V.1. Introducción

Para la mayoría de las ecuaciones con que nos enfrentamos al abordar problemas concretos de ciencia e ingeniería, no nos es posible encontrar una solución exacta de forma analítica. Por ello nos vemos obligados a resolverlas de forma aproximada mediante métodos numéricos. Una herramienta muy general para resolver problemas que involucre la resolución de ecuaciones en derivadas parciales es el FEM. El cual está basado en la reformulación del problema en forma variacional y la solución del espacio de soluciones por otro de dimensión finita. Esto se logra al realizar una partición del dominio en elementos de tamaño finito (segmentos en una dimensión, triángulos y tetraedros en dos o tres dimensiones respectivamente, pero también con cualquier otra forma geométrica con la que se pueda hacer una partición adecuada

del dominio) y se busca soluciones mediante familias de funciones dependientes de un número finito de parámetros (como pueden ser polinomios a trozos de primer grado o de un mayor grado o también se puede utilizar familias de funciones más generales que tienen mejores propiedades de aproximación y la probabilidad de reducir el número de cálculos necesarios para obtener la solución). De esta manera pasamos de un problema inicial a un sistema de ecuaciones que se puede resolver numéricamente.

V.2. Descripción del Método de Elementos Finitos

En el FEM se inicia haciendo una reformulación variacional del problema planteado. Para el caso de ecuaciones elípticas (ecuación de Helmholtz), la formulación variacional transforma el problema inicial en uno de minimización:

$$u \in F \mid J(u) \leq J(f) \quad \forall f \in F, \tag{72}$$

donde F es el conjunto de funciones admisibles y J es un funcional. Las funciones $f \in F$ a menudo representan cantidades que varían en forma continua (como puede ser la temperatura, la presión o el desplazamiento de un cuerpo elástico entre otras) y $J(f)$ es la energía total asociada a f. Por lo que tenemos un problema de minimizar la energía total del sistema. A esta forma de expresar el problema se le conoce como *forma variacional de Ritz*.

Existen otras formas de expresar el problema, la que utilizaremos para este trabajo y que consiste en multiplicar ambos lados de la ecuación (Helmholtz) por una *función test* f arbitraria y adecuada e integrar sobre todo el dominio del problema. Aplicando el teorema de Green podemos convertirla en una ecuación integral en la que un miembro no depende de la función incógnita. A esta forma de expresar el problema se le conoce

como *forma variacional de Gakerkin*, de *Ritz-Galerkin* o *forma débil*. En general el espacio de funciones F tiene dimensión infinita, lo que nos imposibilita resolver el problema de manera analítica. Lo que se hace es sustituir F por F_h, donde F_h depende de un número finito de parámetros. Utilizando la *forma variacional de Ritz*, el problema queda convertido en

$$u_h \in F_h \mid J(u_h) \leq J(f) \quad \forall f \in F_h, \tag{73}$$

o bien en un sistema de ecuaciones algebraicas (*forma débil*) en u_h

$$u_h \in F_h \mid J(u_h, f_h) = G(f_h). \tag{74}$$

Resolver un problema mediante el método de elementos finitos se puede sintetizar en:

- Formulación variacional del problema. Esto se realiza multiplicando la ecuación por una función de prueba f, integrando y aplicando el teorema de Green.
- Generar una partición del dominio en elementos de tamaño finito (utilizando cualquier forma geométrica con la que se pueda hacer una partición adecuada del dominio).
- Crear el espacio F_h (en función de la partición que se creó en el punto anterior). Esto implica buscar familias de funciones dependientes de un número finito de parámetros (como puede ser polinomios a trozos de primer grado o de un grado superior, etc).
- Solución del problema discreto. Esto se reduce a resolver el sistema de ecuaciones algebraico.

V.3. Espacio de Sobolev

Es un subespacio vectorial del espacio L^p formado por clases de funciones tales que sus derivadas hasta orden n pertenecen también a L^p.

Sea Ω un dominio en $\mathbb{R}^n$ y p un número real. Denotamos por $L^p(\Omega)$ el espacio de todas las funciones medibles f definidas en Ω que cumplen:

$$\int_{\Omega} \mid f(x) \mid^p < \infty. \tag{75}$$

Para $1 \leq p \leq \infty$ definimos la norma como

$$\parallel f \parallel_{L^p(\Omega)} = \left(\int_{\Omega} \mid f(x) \mid^p dx \right)^{\frac{1}{p}} \tag{76}$$

y para que $p \to \infty$,

$$\parallel f \parallel_{L^\infty(\Omega)} = \sup \mid f(x) \mid : x \in \Omega. \tag{77}$$

Con estas condiciones se define los espacios de Lebesgue como :

$$L^p(\Omega) = \left\{ f : f \text{ está definido en } \Omega \text{ y } \parallel f \parallel_{L^\infty(\Omega)} < \infty \right\}. \tag{78}$$

Usando la notación:

$$D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_1^{\alpha_n}} \quad \text{con} \quad \mid \alpha \mid = \alpha_1 + \cdots + \alpha_n, \tag{79}$$

para indicar la derivada parcial de f.

En el FEM no hace falta utilizar la derivada puntual sino en términos de integración, lo que nos lleva a introducir el concepto de derivada débil. Una función $f \in L^1_{loc}(\Omega)$

tiene derivada débil $D^{\alpha}u = f_{\alpha}$ si existe una función $u \in L^1_{loc}(\Omega)$ tal que

$$\int_{\Omega} u(x) D^{\alpha}\phi(x)dx = (-1)^{|\alpha|} \int_{\Omega} f_{\alpha}(x)\phi(x)dx, \tag{80}$$

para cada $\phi \in D(\Omega)$. Entendiendo que si no existe la derivada en sentido tradicional, nos estaremos refiriendo a la derivada débil.

Para generalizar los espacios de Lebesgue introduciremos la norma de Sobolev:

$$\| u \|_{Z^{m.p}(\Omega)} = \left(\sum_{0 \leq |\alpha| \leq m} \| D^{\alpha}u \|^p_{L^p(\Omega)} \right)^{1/p} \quad \text{si} \quad 1 \leq p < \infty, \tag{81}$$

$$\| u \|_{Z^{m.\infty}(\Omega)} = \max_{0 \leq |\alpha| \leq m} \| D^{\alpha}u \|_{L^{\infty}(\Omega)} \quad \text{si} \quad p \to \infty. \tag{82}$$

Los espacios de Sobolev se definen como

$$Z^{m.p}(\Omega) = \{ f \in L^1_{loc}(\Omega) : \| f \|_{Z^{m.p}(\Omega)} < \infty \}. \tag{83}$$

Ahora tomando $p = 2$, se establecen los espacios H^m en la forma

$$H^m = \{ f \in L^1_{loc} : \| f \|_{Z^{m,2}} < \infty, \forall p \leq m \}, \tag{84}$$

donde H^m es la familia de funciones f tales que tanto ellas como todas sus derivadas $D^{\alpha f}$ hasta orden m son funciones de cuadrado integrable en Ω. Utilizando estas condiciones H^m, con el producto escalar

$$(u, f)_{H^m(\Omega)} = \sum_{0 \leq |\alpha| \leq m} \int_{\Omega} D^{\alpha}u(x) D^{\alpha}f(x)dx, \quad u, f \in H^m(\Omega) \tag{85}$$

y la norma correspondiente $\| \cdot \|_{H^m(\Omega)}$, es un espacio de Hilbert (Adams and Fournier, 2003; Brezis, 2011).

V.4. Teorema de Lax-Milgram

El teorema de Lax-Milgram se presentó por primera vez en 1954, en *Parabolic equations* (Lax and Milgram, 1954). En un inicio se introdujo como un lema para probar un resultado sobre ecuaciones en derivadas parciales parabólicas y debido a su importancia cada vez se le considera más como "teorema". Este teorema utiliza el método variacional para darnos existencia y unicidad en el sentido débil de diferentes problemas de contorno que involucran EDPs, en especial las de tipo elíptico.

Lema V.4.1. *Sean X, Y espacios normados. Entonces una aplicación lineal $L : X \to Y$ admite una inversa continua si y solamente si L está acotada inferiormente.*

Lema V.4.2. *Sea X un espacio de Banach y $L : X \to X$ un operador lineal continuo. Si L está acotado inferiormente, entonces $R(L)$, la imagen de L, es cerrada en X.*

Teorema V.4.1. *Sea M un subespacio cerrado apropiado de un espacio de Banach X. Entonces, para todo $x_0 \in X \setminus M$ existe un funcional lineal continuo $f_0 \in X^*$ tal que $f_0(x_0)$ y $f_0 = 0$ en M.*

Teorema V.4.2. (Teorema de Lax-Milgram). *Sea H un espacio de Hilbert y $a : H \times H \to \mathbb{R}$ una forma bilineal con las siguientes propiedades:*

1. a es continua: existe una constaste real C tal que

$$|a(x,x)| \leq C \parallel x \parallel \parallel y \parallel, \quad \forall x, y \in H. \tag{86}$$

2. a es coerciva: existe una constaste real positiva α tal que

$$|a(x,x)| \geq \alpha \parallel x \parallel^2, \quad \forall x \in H. \tag{87}$$

Entonces, para cada $f \in H^*$ existe un único $y_0 \in H$ tal que $a(x, y_0) = f(x), \forall x \in H$.

Demostración: Para cada $y \in H$ fijo, se tiene que $a(\cdot\,, y)$: $H \to \mathbb{R}$ es un funcional lineal

continuo. Se puede definir la aplicación $A : H \to H^*$ dada por $Ay = a(\cdot\,, y)$. A está acotada inferiormente

$$\alpha \quad \| \quad y \|^2 \leq | \, a(y,y) \, | = | \, Ay(y) \, | = \| \, y \, \| \, Ay \left(\frac{y}{\| \, y \, \|} \right) \tag{88}$$

$$\leq \quad \| \, y \, \| \sup_{\|y\|=1} \{ Ay \, (y) \} = \| \, y \, \| \| \, Ay \, \|, \quad \forall y \in H, \tag{89}$$

con lo cual nos queda que

$$\alpha \, \| \, y \, \| \leq \| \, Ay \, \|, \quad \forall y \in H. \tag{90}$$

Por los lemas V.4.1, V.4.2 tenemos que A es inyectiva y que $R(A)$ es cerrada en H^*.

Si la imagen de A es H^*, se supone lo contrario y llegamos a una contradicción. Además, se supone que existe $\hat{f} \neq 0 \in H^*$ tal que $\hat{f} \notin R(A)$. Por el teorema Teorema V.4.1, existe $z \in H^{**}$ tal que $z(\hat{f}) > 1$ y $z(f) = 0$ para cualquier $f \in R(A)$. Como H es reflexivo, por ser un espacio de Hilbert, podemos identificar z con $x \in H$, por lo que las propiedades de z se transforman en $\hat{f}(x) > 1$ y $f(x) = a(x,y) = Ay(x) = 0$ para cada $f \in R(a)$ y para cada $y \in H$. La coercividad de a y la segunda de estas dos propiedades implica que $x = 0$ y por la linealidad $\hat{f}$, $\hat{f}(x) = 0$, lo cual es una contradicción y, por lo tanto $R(A) = H^*$.

Tenemos que para cada $f \in H^*$, existe $y_0 \in H$ tal que $f = Ay_0$; es decir, $f(x) = Ay_0(x) = a(x, y_0)$ para todo x en H. Como A es inyectiva y_0 es único, lo cual es justo lo que se quería demostrar.

Las demostraciones a los lemas V.4.1, V.4.2, al teorema V.4.1 y más información relacionada al teorema de Lax-Milgram se puede encontrar en (Brezis, 2011; Álvarez Melis, 2011; Pérez, 2017).

V.5. Discretización del dominio F_h

Iniciaremos con la discretización de un problema en una dimensión y posteriormente se abordará en dos y tres dimensiones. Supongamos que el dominio es $[x_0, x_M]$. Hacemos una partición del dominio de la forma $x_0 < x_1 < x_2 < \ldots < x_{M-2} < x_{M-1} < x_M$ y designamos con $h_j = x_j - x_{j-1}$ y $h = \max(h_j)$.

Sea $F_h = \{f | f$ es afín en cada intervalo de la partición, continua en $[x_0, x_M]$, $f(x_0) = g_1$ y $f(x_M) = g_2\}$. Consideremos la base formada por las funciones sombrero (ver Fig. 10), afines en cada intervalo de la partición y que cumplan:

$$\varphi_j(x_k) \begin{cases} 0 & \text{si } k \neq j \\ 1 & \text{si } k = j, \end{cases} \tag{91}$$

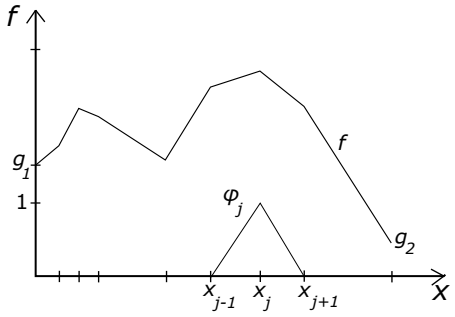

Figura 10: Función f y la función de forma φ_j.

que con esto podemos expresar de manera única,

$$f(x) = \sum_{i=1}^{M} \eta_i \varphi_i(x). \tag{92}$$

Con esto se ha logrado reemplazar el espacio de funciones F por otro $F_h \subset F$ de dimensión finita N. Para el caso en dos dimensiones, la discretización se hace de

manera análoga al caso de una dimensión, salvo que para el plano se tiene más libertad a la hora de elegir la geometría de la partición; se puede utilizar triángulos, rectángulos, hexágonos u otras figura geométricas.

El vector del espacio F_h puede ser expresado de la forma

$$u(x, y) = u_0 + u_x x + u_y y, \tag{93}$$

o

$$u(x, y) = u_0 + u_x x + u_y y + u_{xx} x^2 + u_{xy} xy + u_{yy} y^2. \tag{94}$$

En el primer caso tenemos tres grados de libertad, estos pueden elegirse determinando la función en cada vértice o eligiendo la función en el punto medio de cada lado. Para el segundo caso se tiene seis grados de libertad, los cuales vienen de considerar al espacio F_h formado por funciones cuadráticas a trozos. Para expresar una función de F_h en base $\{u_0, u_x, u_y, u_{uu}, u_{xy}, u_{yy}\}$ es necesario conocer el valor en seis puntos; como puede ser el valor de la función en cada vértice y en los centros de cada lado o bien el valor de la función y de su derivada en cada uno de los vértices, etc. De forma análoga podemos tratar el caso en tres dimensiones; esto nos lleva a tener una gran variedad de elementos para hacer la partición del espacio. Una de las más comunes es utilizar el tetraedro. Para este caso podemos proponer una aproximación lineal para u de la siguiente manera:

$$u(x, y, z) = u_0 + u_x x + u_y y + u_z z, \tag{95}$$

o bien $\mathbf{u} = (u_0, u_x, u_y, u_z)$. En este caso tenemos cuatro grados de libertad y pueden elegirse determinando el valor de la función en los cuatro vértices del tetraedro. En el siguiente capítulo aplicaremos el FEM a cierto tipo de problemas básicos para verificar el método numérico.

Capítulo VI

MÉTODO DE ELEMENTOS FINITOS APLICADO A LA ECUACIÓN DE HELMHOLTZ

En este capítulo se utilizará el Método de Elementos Finitos para resolver la ecuación de Helmholtz en una y dos dimensiones con dominios finitos. Para esto se combinará el Método de Elementos Finitos con la técnica PML.

VI.1. Técnica PML

Un enfoque alternativo para lidiar con el truncamiento de dominios ilimitados es utilizar la técnica conocida como "Perfectly Mathed Layer" (PML) (Berenger, 1994, 1996). Ésta se basa en simular una capa de material absorbente que se utiliza para rodear el dominio de interés. La propiedad clave que distingue al material utilizado en esta técnica con uno ordinario es que éste está diseñado para que las ondas que inciden desde el dominio de interés sean absorbidas y así evitar que sean reflejadas. La técnica PML fue propuesta por Berenger en 1994 (Berenger, 1994) para su uso en las ecuaciones de Maxwell. La propuesta original de Berenger se denominaba PML de *campo dividido*, esto se debe a que en la región de la PML el campo electromagnético era dividido

en dos campos. Posteriormente se formuló una versión más simple y eficiente que fue llamada PML *uniaxial* (Gedney, 1997). En esta formulación la PML es descrita como un material absorbente anisotrópico artificial. Ambas formulaciones parten construyendo manualmente las condiciones bajo las cuales las ondas planas incidentes no reflejan desde el material PML. Posteriormente se demostró que las PMLs corresponden a una transformación de coordenadas, donde una o más de las coordenadas se asignan a números complejos. Esto es en realidad una continuación analítica de la ecuación de onda en coordenadas complejas, que reemplaza las ondas de oscilantes por ondas de decaimiento exponencial.

La técnica PML, en su forma original, sólo atenúan las ondas de propagación y las ondas evanescentes oscilan en la PML con una atenuación, pero esto puede ser resulto al incluir un estiramiento de coordenadas reales, lo cual corresponde a hacer σ una expresión compleja.

VI.2. Caso 1D

Consideremos el siguiente problema:

$$\bigtriangledown^2 u + k^2 u = 0, \tag{96}$$

donde k es el número real de onda y D es un dominio infinito con frontera interior $\partial D = \partial D_D \cup \partial D_N$, sujeto a las condiciones de frontera de Dirichlet en ∂D_D,

$$u \mid_{\partial D_D} = u_0, \tag{97}$$

condiciones de frontera de Neumann en ∂D_N,

$$\frac{\partial u}{\partial n} \mid_{\partial D_N} = g_0, \tag{98}$$

y/o la condición de radiación de Sommerfeld,

$$\lim_{r\to\infty} r^{1/2}\left(\frac{\partial u}{\partial r} - iku\right) = 0. \tag{99}$$

Se quiere resolver el sistema de Ecs. (96)-(99) en un dominio finito mediante FEM, aplicando la condición de radiación de Sommerfeld y la técnica PML. Para ello, apliquemos el mapeo de coordenadas complejas a la ecuación de Helmholtz (Teixeira and Chew, 1998):

$$\frac{\partial}{\partial x} \rightarrow \frac{1}{\gamma(x)}\frac{\partial}{\partial x}, \tag{100}$$

y multiplicando por γ se obtiene:

$$\frac{\partial}{\partial x}\left(\frac{1}{\gamma}\frac{\partial u}{\partial x}\right) + k^2\gamma u = 0. \tag{101}$$

Rodeamos el dominio de interés D_c por una región PML de ancho δ_{PML} y elegimos γ como

$$\gamma = \begin{cases} 1 & \text{en } D_c \\ 1 + \frac{i}{k}\sigma(\nu) & \text{en la region PML,} \end{cases} \tag{102}$$

donde ν es la distancia de un punto en la región PML a la frontera D_c. Recientemente Bermúdez (Bermúdez *et al.*, 2007) propuso funciones absorbentes de la forma

$$\sigma = \frac{c}{\delta_{PML} - \nu}, \tag{103}$$

que son singulares en el límite exterior de la región PML.

VI.2.1. Solución a la ecuación de Helmholtz 1D en forma general

Como vimos en el capítulo anterior se tiene que seguir una serie de pasos para resolver el problema utilizando FEM.

Generar una partición

Para tal fin se definirá el dominio $D = D_c \cup D_{PML}$ del problema. Con $D_c = \{x \mid x \in [x_1, x_{PML}]\}$ y $D_{PML} = \{x \mid x \in [x_{PML}, x_{PML} + \delta_{PML}]\}$ y donde δ_{PML} es el espesor de la capa de material absorbente.

Dividamos el dominio físico en subdominios o elementos finitos como se muestra en la Fig. 11.

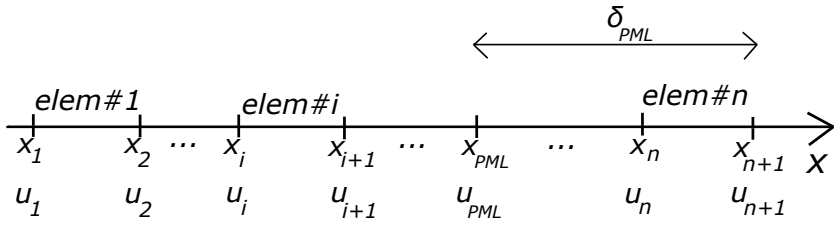

Figura 11: División del dominio físico D en N elementos y en $N + 1$ nodos.

Crear el espacio de funciones u_h

Definamos una expresión para u en términos de u_i y u_{i+1}

$$u = H_1(x)u_i + H_2(x)u_{i+1}, \tag{104}$$

donde

$$H_1(x) = \frac{x_{i+1} - x}{h_i}, \tag{105}$$

$$H_2(x) = \frac{x - x_i}{h_i}, \tag{106}$$

$$h_i = x_{i+1} - x_i, \tag{107}$$

son funciones lineales de forma, las cuales tienen las siguientes propiedades:

- La función de forma asociada al nodo i tiene el valor de 1 en el nodo i y 0 en cualquier otro nodo,

$$H_1(x_i) = 1,\ H_1(x_{i+1}) = 0,\ H_2(x_i) = 0,\ H_2(x_{i+1}) = 1. \tag{108}$$

- La suma de todas las funciones de forma es la unidad,

$$\sum_{u=1}^{2} H_i(x) = 1. \tag{109}$$

Formulación de Galerkin en su forma débil

En esta sección utilizaremos la formulación de Galerkin en su forma débil aplicado a la Ec. (101). Para esto transformaremos la ecuación antes mencionada a su forma débil,

$$\begin{aligned} I &= \sum_{i=1}^{PML-1} \int_{x_i}^{x_{i+1}} w \left(\frac{d^2u}{dx^2} + k^2 u \right) dx \\ &\quad + \sum_{i=PML}^{N} \int_{x_i}^{x_{i+1}} w \left(\frac{d}{dx} \frac{1}{\gamma} \frac{du}{dx} + k^2 \gamma u \right) dx \\ I &= \sum_{i=1}^{PML-1} \int_{x_i}^{x_{i+1}} \left(\frac{dw}{dx} \frac{du}{dx} - k^2 w u \right) dx \\ &\quad + \sum_{i=PML}^{N} \int_{x_i}^{x_{i+1}} \left(\frac{1}{\gamma} \frac{dw}{dx} \frac{du}{dx} - k^2 \gamma w u \right) dx - \left[w \frac{du}{dx} \right]_{x_i}^{x_f} = 0. \end{aligned} \tag{110}$$

Sustituyendo la Ec. (104) y la función de prueba $w(w_1(x), w_2(x))$ por $w(H_1(x), H_2(x))$ en la Ec. (110) obtenemos

$$\sum_{i=1}^{PML-1}\int_{x_i}^{x_{i+1}}\left(\begin{Bmatrix}H_1'\\H_2'\end{Bmatrix}[H_1'H_2'] - k^2\begin{Bmatrix}H_1\\H_2\end{Bmatrix}[H_1H_2]\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix} + \sum_{i=PML}^{N}\int_{x_i}^{x_{i+1}}\left(\frac{1}{\gamma}\begin{Bmatrix}H_1'\\H_2'\end{Bmatrix}[H_1'H_2'] - k^2\gamma\begin{Bmatrix}H_1\\H_2\end{Bmatrix}[H_1H_2]\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix} = \left[w\frac{du}{dx}\right]_{x_i}^{x_f}, \tag{111}$$

donde H_j' denota $\frac{dH_j(x)}{dx}$. La Ec. (111) contiene las dos regiones que componen el domino D, la región que representa el dominio físico o de interés D_c y la región PML D_{PML}.

Solución del problema

La Ec. (111) genera un sistema de ecuaciones lineales, para el cual la mayoría de los elementos de la matriz son nulos. Algunos de los métodos que se puede utilizar para resolver el sistema pueden ser: LU, Gradiente conjugado, Cholesky, GMRES, entre muchos otros.

VI.2.2. Solución a la ecuación de Helmholtz 1D para un problema particular

Ya con las ecuaciones generales definidas podemos pasar a un caso particular del problema (Ec.(96)). Para esto definamos $\delta_{PML} = 0.5$, $D = \{x|x \in [0, 1.5]\}$, $D_c = \{x|x \in [0, 1]\}$, $D_{PML} = \{x|x \in [1, 1.5]\}$, $k = 8\pi$ y el dominio D se dividirá en seis elementos y siete nodos como se muestra en la Fig. (12). Para facilitar el problema utilizaremos las condiciones de Dirichlet, con $u(0) = 1$ y $u(1.5) = 0$.

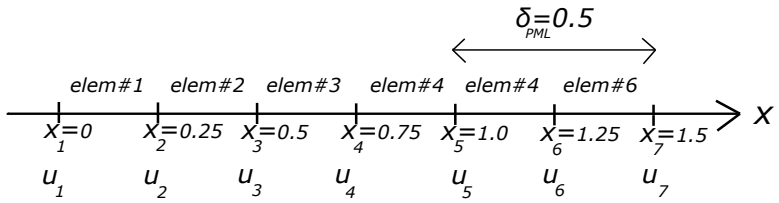

Figura 12: División del dominio físico D en 6 elementos y 7 nodos.

Adaptando la Ec. (111) a nuestro problema particular ésta se transforma en

$$\sum_{i=1}^{4}\int_{x_i}^{x_{i+1}}\left(-\begin{Bmatrix}H_1'\\H_2'\end{Bmatrix}\{H_1'H_2'\}+k^2\begin{Bmatrix}H_1\\H_2\end{Bmatrix}\{H_1H_2\}\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix}+$$
$$\sum_{i=5}^{6}\int_{x_i}^{x_{i+1}}\left(-\frac{1}{\gamma}\begin{Bmatrix}H_1'\\H_2'\end{Bmatrix}\{H_1'H_2'\}+k^2\gamma\begin{Bmatrix}H_1\\H_2\end{Bmatrix}\{H_1H_2\}\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix}=0, \quad (112)$$

donde

$$Ke=\begin{cases}\displaystyle\int_{x_i}^{x_{i+1}}\left(-\begin{Bmatrix}H_1'\\H_2'\end{Bmatrix}\{H_1'H_2'\}\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix} & i\leq 4\\ \displaystyle\int_{x_i}^{x_{i+1}}\left(-\frac{1}{\gamma}\begin{Bmatrix}H_1'\\H_2'\end{Bmatrix}\{H_1'H_2'\}\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix} & i>4\end{cases} \quad (113)$$

y

$$Me=\begin{cases}\displaystyle\int_{x_i}^{x_{i+1}}\left(k^2\begin{Bmatrix}H_1\\H_2\end{Bmatrix}\{H_1H_2\}\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix} & i\leq 4\\ \displaystyle\int_{x_i}^{x_{i+1}}\left(k^2\gamma\begin{Bmatrix}H_1\\H_2\end{Bmatrix}\{H_1H_2\}\right)dx\begin{Bmatrix}u_i\\u_{i+1}\end{Bmatrix} & i>4.\end{cases} \quad (114)$$

Calculando las integrales para cada elemento tenemos que

Elemento #1

$$Ke=\begin{bmatrix}-4 & 4\\ 4 & -4\end{bmatrix}\begin{Bmatrix}u_1\\u_2\end{Bmatrix} \quad \text{y} \quad Me=\begin{bmatrix}52.6379 & 26.3189\\ 26.3189 & 52.6379\end{bmatrix}\begin{Bmatrix}u_1\\u_2\end{Bmatrix}. \quad (115)$$

Elemento #2

$$Ke = \begin{bmatrix} -4 & 4 \\ 4 & -4 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} \quad \text{y} \quad Me = \begin{bmatrix} 52.6379 & 26.3189 \\ 26.3189 & 52.6379 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix}. \tag{116}$$

Elemento #3

$$Ke = \begin{bmatrix} -4 & 4 \\ 4 & -4 \end{bmatrix} \begin{Bmatrix} u_3 \\ u_4 \end{Bmatrix} \quad \text{y} \quad Me = \begin{bmatrix} 52.6379 & 26.3189 \\ 26.3189 & 52.6379 \end{bmatrix} \begin{Bmatrix} u_3 \\ u_4 \end{Bmatrix}. \tag{117}$$

Elemento #4

$$Ke = \begin{bmatrix} -4 & 4 \\ 4 & -4 \end{bmatrix} \begin{Bmatrix} u_4 \\ u_5 \end{Bmatrix} \quad \text{y} \quad Me = \begin{bmatrix} 52.6379 & 26.3189 \\ 26.3189 & 52.6379 \end{bmatrix} \begin{Bmatrix} u_4 \\ u_5 \end{Bmatrix}. \tag{118}$$

Elemento #5

$$\begin{aligned} Ke &= \begin{bmatrix} -3.9501 + 0.4353i & 3.9501 - 0.4353i \\ 3.9501 - 0.4353i & -3.9501 + 0.4353i \end{bmatrix} \begin{Bmatrix} u_5 \\ u_6 \end{Bmatrix} \quad \text{y} \\ Me &= \begin{bmatrix} 52.6379 + 4.8542i & 105.2743 + 10.5693i \\ 105.2743 + 10.5693i & 368.4560 + 43.6864i \end{bmatrix} \begin{Bmatrix} u_5 \\ u_6 \end{Bmatrix}. \end{aligned} \tag{119}$$

Elemento #6

$$\begin{aligned} Ke &= \begin{bmatrix} -3.1006 + 1.1780i & 3.1006 - 1.1780i \\ 3.1006 - 1.17801i & -3.1006 + 1.1780i \end{bmatrix} \begin{Bmatrix} u_6 \\ u_7 \end{Bmatrix} \quad \text{y} \\ Me &= \begin{bmatrix} 52.6379 + 12.5653i & 9.5708 + 200.9444i \\ 109.5708 + 200.9444i & 355.5068 + 440.1354i \end{bmatrix} \begin{Bmatrix} u_6 \\ u_7 \end{Bmatrix}. \end{aligned} \tag{120}$$

Como podemos observar cada elemento tiene diferentes nodos asociados con él, si

expandimos cada ecuación de modo que se tenga una matriz y un vector del tamaño de los grados de libertad asociado al sistema. Para nuestro caso tenemos siete grados de libertad, que con esto en cuenta podemos reescribir de las Ecs. (115) - (120) como

Elemento #1

$$Ke = \begin{Bmatrix} -4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 4 & -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} \text{ y } Me = \begin{Bmatrix} 52.6379 & 26.3189 & 0 & 0 & 0 & 0 & 0 \\ 26.3189 & 52.6379 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \tag{121}$$

Elemento #2

$$Ke = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 4 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} \text{ y } Me = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 52.6379 & 26.3189 & 0 & 0 & 0 & 0 \\ 0 & 26.3189 & 52.6379 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \tag{122}$$

Elemento #3

$$Ke = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & 4 & 0 & 0 & 0 \\ 0 & 0 & 4 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} \text{ y } Me = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 52.6379 & 26.3189 & 0 & 0 & 0 \\ 0 & 0 & 26.3189 & 52.6379 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \tag{123}$$

Elemento #4

$$Ke = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} \text{ y } Me = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 52.6379 & 26.3189 & 0 & 0 \\ 0 & 0 & 0 & 26.3189 & 52.6379 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \tag{124}$$

Elemento #5

$$\begin{aligned} Ke &= \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3.9501 + 0.4353i & 3.9501 - 0.4353i & 0 \\ 0 & 0 & 0 & 0 & 3.9501 - 0.4353i & -3.9501 + 0.4353i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} \text{ y} \\ Me &= \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 52.6379 + 4.8542i & 105.2743 + 10.5693i & 0 \\ 0 & 0 & 0 & 0 & 105.2743 + 10.5693i & 368.4560 + 43.6864i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \end{aligned} \tag{125}$$

Elemento #6

$$Ke = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3.1006 + 1.1780i & 3.1006 - 1.1780i \\ 0 & 0 & 0 & 0 & 0 & 3.1006 - 1.17801i & -3.1006 + 1.1780i \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} \text{ y}$$

$$Me = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 52.6379 + 12.5653i & 9.5708 + 200.9444i \\ 0 & 0 & 0 & 0 & 0 & 109.5708 + 200.9444i & 355.5068 + 440.1354i \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \quad (126)$$

Sumando las matrices Ke y Me de cada elemento se obtiene la matriz de rigidez K y la de masas M como se muestra enseguida:

$$K = \begin{Bmatrix} -4.0 + 0.0i & 4.0 + 0.0i & 0 & 0 & 0 \\ 4.0 + 0.0i & -8.0 + 0.0i & 4.0 + 0.0i & 0 & 0 \\ 0 & 4.0 + 0.0i & -8.0 + 0.0i & 4.0 + 0.0i & 0 \\ 0 & 0 & 4.0 + 0.0i & -8.0 + 0.0i & 4.0 + 0.0i \\ 0 & 0 & 0 & 4.0 + 0.0i & -7.9501 + 0.4353i \\ 0 & 0 & 0 & 0 & 3.9501 - 0.4353i \\ 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

$$\begin{Bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 3.9501 - 0.4353i & 0 \\ -7.0507 + 1.6133i & 3.1006 - 1.1780i \\ 3.1006 - 1.17801i & -3.1006 + 1.1780i \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}, \quad (127)$$

$$M = \begin{Bmatrix} 52.6378 + 0.0i & 26.3189 + 0.0i & 0 & 0 & 0 \\ 26.3189 + 0.0i & 105.2757 + 0.0i & 26.3189 + 0.0i & 0 & 0 \\ 0 & 26.3189 + 0.0i & 105.2757 + 0.0i & 26.3189 + 0.0i & 0 \\ 0 & 0 & 26.3189 + 0.0i & 105.2757 + 0.0i & 26.3189 + 0.0i \\ 0 & 0 & 0 & 26.3189 + 0.0i & 105.2758 + 4.8542i \\ 0 & 0 & 0 & 0 & 105.2743 + 10.5693i \\ 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

$$\begin{Bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 105.2743 + 10.5693i & 0 \\ 421.0940 + 56.2518i & 109.5708 + 200.9444i \\ 109.5708 + 200.9444i & 355.5068 + 440.1354i \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \tag{128}$$

Una vez obtenidas las matrices K y M se procede a sumarlas

$$(K+M) = \begin{Bmatrix} 48.6378 + 0.0i & 30.3189 + 0.0i & 0 & 0 & 0 \\ 30.3189 + 0.0i & 97.2757 + 0.0i & 30.3189 + 0.0i & 0 & 0 \\ 0 & 30.3189 + 0.0i & 97.2757 + 0.0i & 30.3189 + 0.0i & 0 \\ 0 & 0 & 30.3189 + 0.0i & 97.2757 + 0.0i & 30.3189 + 0.0i \\ 0 & 0 & 0 & 30.3189 + 0.0i & 97.3257 + 5.2895i \\ 0 & 0 & 0 & 0 & 109.2244 + 10.1340i \\ 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

$$\begin{Bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 109.2244 + 10.1340i & 0 \\ 414.0433 + 57.8651i & 112.6714 + 199.7664i \\ 112.6714 + 199.7664i & 352.4062 + 441.3134i \end{Bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix}. \tag{129}$$

Ya sumadas la matriz de rigideces y masas, se procede a agregar la condiciones de Dirichlet. Para esto simplemente hay que notar que en la matriz $(K+M)$ contiene el nodo u_1 cuyo valor es conocido por la condición de Dirichlet. Para más información de como agregar la condición de Dirichlet consultar la Ref. (Kwon and Bang, 1996). Así,

$$\left\{\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 97.2757+0.0i & 30.3189+0.0i & 0 & 0 \\ 0 & 30.3189+0.0i & 97.2757+0.0i & 30.3189+0.0i & 0 \\ 0 & 0 & 30.3189+0.0i & 97.2757+0.0i & 30.3189+0.0i \\ 0 & 0 & 0 & 30.3189+0.0i & 97.3257+5.2895i \\ 0 & 0 & 0 & 0 & 109.2244+10.1340i \\ 0 & 0 & 0 & 0 & 0 \end{array}\right.$$

$$\left.\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 109.2244+10.1340i & 0 \\ 414.0433+57.8651i & 112.6714+199.7664i \\ 112.6714+199.7664i & 352.4062+441.3134i \end{array}\right\} \left\{\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{array}\right\} + \left\{\begin{array}{c} 1.0+0.0i \\ 30.3189+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \end{array}\right\} = 0, \tag{130}$$

donde

$$F = (-1)\left\{\begin{array}{c} 1.0+0.0i \\ 30.3189+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \\ 0.0+0.0i \end{array}\right\}. \tag{131}$$

La Ec. (130) puede ser reescrita como

$$(K+M)\cdot u = F, \tag{132}$$

donde $u = (u_1, u_2, u_3, u_4, u_5, u_6, u_7)$. La solución a la Ec. (132) es:

$$u = \left\{ \begin{array}{c} 1.0000 + 0.0000i \\ -0.3500 + 0.0001i \\ 0.1229 - 0.0002i \\ -0.0444 + 0.0005i \\ 0.0196 - 0.0013i \\ -0.0051 + 0.0006i \\ 0.0000 + 0.0000i \end{array} \right\} \tag{133}$$

y la parte real de la solución analítica a nuestro problema inicial en el dominio de interés D_c es

$$u = \cos(8\pi x). \tag{134}$$

En la Fig. 13 y en la Tabla III se observa que el cálculo mediante el FEM aumenta en precisión al aumentar el número elementos en los que se encuentra divido el espacio D_c.

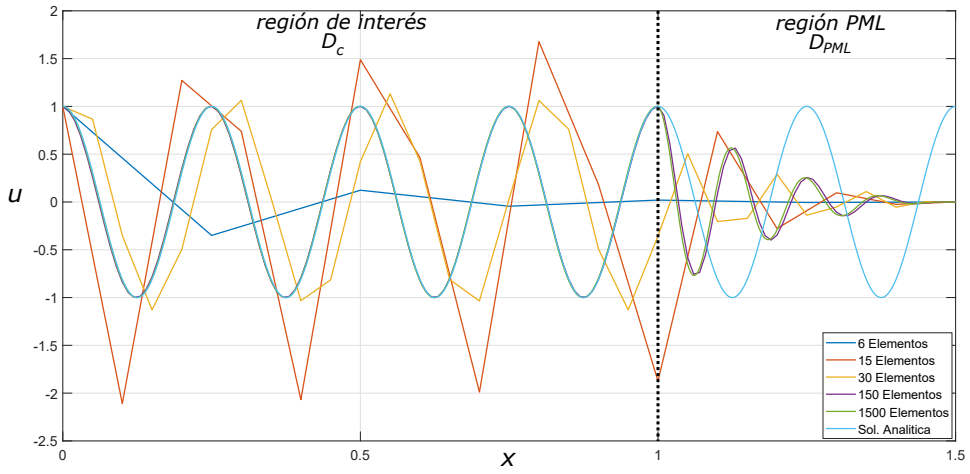

Figura 13: Solución analítica y numérica de la Ec. (96) variando el número de elementos que conforman el dominio de interés D.

Tabla III: Resultados numéricos y error relativo variando el número de elementos.

x	Sol. Analítica	Sol. Num. 15 elem.	Error Relativo 15 elem.(%)	Sol. Num. 30 elem.	Error Relativo 30 elem.(%)	Sol. Num. 150 elem.	Error Relativo 150 elem.(%)	Sol. Num. 1500 elem.	Error Relativo 1500 elem.(%)
0.0	1.0	1.0	0	1.0	0	1.0	0	1.0	0
0.1	-0.8090	-2.1094	-160.73	-0.3505	-56.67	-0.8384	-3.6258	-0.8225	-1.6723
0.2	0.3090	1.2720	311.62	-0.4961	260.54	0.3500	13.2551	0.3308	7.0632
0.3	0.3090	0.7393	139.23	1.0637	244.23	0.2748	11.0762	0.2873	7.0421
0.4	-0.8090	-2.0683	-155.65	1.0637	-27.71	-0.7925	-2.0459	-0.7956	-1.6571
0.5	1	1.4885	48.84	0.4217	57.83	1.0013	0.1299	1.0000	0.0008
0.6	-0.8090	0.4651	-157.48	0.4270	-152.77	-0.8199	-1.3459	-0.8224	-1.6490
0.7	0.3090	-1.9894	743.77	-1.0355	435.10	0.3190	3.2229	0.3305	6.9622
0.8	0.3090	1.6777	442.92	1.0618	243.59	0.3063	0.8906	0.2876	6.9396
0.9	-0.8090	0.1823	-122.53	-0.4910	-39.31	-0.8121	-0.3870	-0.7958	-1.6323
1.0	1	-1.8741	287.40	-0.3559	135.59	1.0015	0.1519	1.0000	0.0015

Programa 5: Helmholtz 1D

En esta sección se mostrará el programa que utiliza el FEM en combinación con la técnica PML para resolver la Ecuación de Helmholtz 1D.

****

Programa 5. Método de Elementos Finitos en combinación con la técnica PML para la Ecuación de Helmholtz 1D.

****

```
function [U,p]=Helmholtz_1D_Con_PML_V1(Dci,Dcf,Dpml,g,k0,tam)
     % Dci x inicial de la regi\on de interes
     % Dcf x final de la regi\'on de interes
     % tam Tama\~no de los elementos

     pmlx=Dpml; % Tama\~no de la capa PML
     disx=Dc; % Final de la regi\'on de interes Dc
     dpmlx=pmlx+0.00001; % Suma 0.00001 para evitar divici\'on entre
                         % cero

     p=Dci:tam:Dcf+Dpml;p=p(:); % lista las coordenadas x de los N
                                % nodos
     N=length(p); % N\'umero de nodos
     T=N-1; % N\'umero de elementos
     t=[1:(N-1);2:N]'; % t lista los 2 n\'umeros de nodo de los
                       % elementos Tri\'angulos
     b=[1 N]; % Frontera

     K=sparse(N,N); % Matriz de rigideces
     M=sparse(N,N); % Matriz de masas
     F=zeros(N,1); % vector de cargas
     tem=sparse(N,1); % Vector que guardara la interaci\'on de la
                      % frontera con la matriz global
     tem1=ones(N,1); % Vector que almacenar\'a la ubicaci\'on de
                     % los nodos interiores
     tem2=zeros(N,1); % Vector para forzar condici'on de  frontera
     Ke=zeros(2,2); % Matriz auxiliar Ke
     Me=zeros(2,2); % Matriz auxiliar Me
%--------------------------------------------------------------------
     % Calculo de las matrices auxiliares Ke, Me
     % y ensamblaje de la matriz de riguideces K y de masas M
     for e=1:T  % Integraci\'on sobre los elementos
```

```
        nodes=t(e,:); % Nodos del elemento e
        if(p(nodes(2))<=1) % Calculo de la matrices Ke y Me para la
                           % regi'on de interes Dc
            Pe=[ones(2,1),p(nodes,:)]; % Matriz 2 por 2 matrix con
                                       % filas=[1 xcorner]
            Area=abs(det(Pe)); % \'Area del elemento e
            Ke=-[1 -1;-1 1]/Area; % Matriz auxiliar Ke para el
                                  % elemento e
            Me=(k0^2)*Area*[2 1;1 2]/6; % Matriz auxiliar Me para
                                        % el elemento e
        else % Calculo de la matrices Ke y Me para la regi'on Dpml
            for i=1:2
                for j=1:2
                    Ke(i,j)=-1*IntegralK1(i,j,p(nodes),disx,...
                     dpmlx,k0); % Matriz auxiliar Ke para el
                             % elemento "e" en la regi\'on Dpml
                    Me(i,j)=(k0^2)*IntegralM1(i,j,p(nodes),...
                     disx,dpmlx,k0); % Matriz auxiliar Me para
                                     % el elemento "e" en la
                                     % regi\'on Dpml
                end % Fin for j
            end % Fin for i
        end % Fin if
%-------------------------------------------------------------------
        % Ensambaje de la matriz de riguideces K y de masas M
        K(nodes,nodes)=K(nodes,nodes)+Ke; % Se suma Ke a las 2 en-
                                        % tradas de la matriz global K
        M(nodes,nodes)=M(nodes,nodes)+Me; % Se suma Ke a la 2 entra-
                                        % das de la matriz global M
    end % Fin for e
    K=K+M; % Suma de matrices globales
%-------------------------------------------------------------------
    % Condici\'on de frontera
    pf=p(b,:); % Puntos frontera
    for i=1:length(b) % Ciclo para agregar condiciones de frontera
        px=pf(i); % Punto de la frontera
        if(px==0) % Condici\'on de Dirichlet el x=0
            tem=tem+g(px).*K(b(i),:)'.*ones(N,1); % Vector que
                     % guarda la interaci\'on de la condici\'on de
                     % frontera con la Matriz Global K
            tem2(g(i),1)=g(px); % Vector para forzar condici\'on de
                                % Condici\'on de Dirichlet en el
                                % vector de cargas F
        end % Fin if
        tem1(g(i),1)=0; % Elementos internos con "1","0" los nodos
```

```
                        % frontera
    end % Fin for i
    F=F-tem; % Suma del vector de interaci\'on de la condici\'on de
           % la frontera con la matriz Global K al vector de cargas
    F=tem1.*F+tem2; % Forzar condici\'on de frontera en el vector de
                    % cargas

    K(b,:)=0; K(:,b)=0; % Ponemos ceros para no modificar los valores
                        % en la frontera
    K(b,b)=speye(length(b),length(b)); % se pone I  en la submatriz
                                       % "frontera" de K
%--------------------------------------------------------------------------
    % Ahora resolvemos
    U=K\F;  % La aproximaci\'on es U_1 phi_1 + ... + U_N phi_N
end % Fin del programa
```

Los algoritmos utilizadas por el programa 5 se pueden encontrar en el apéndice D.

VI.3. Caso 2D

Para abordar problemas en dos dimensiones se hace de manera muy similar al problema en una dimensión que se trató anteriormente. Por esta razón ya no se hará una explicación tan exhaustiva del método, salvo algunas partes relevantes.

Consideremos un problema en $\mathbb{R}^2$ con un obstáculo que evite la propagación de las ondas como se muestra en la Fig. 14.

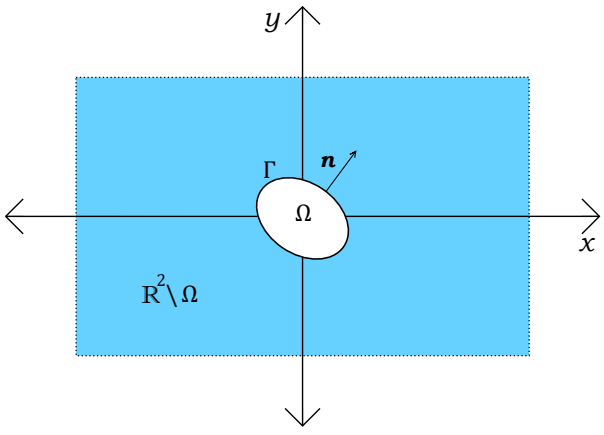

Figura 14: Dominio bidimensional ilimitado.

Nuestro objetivo es resolver el siguiente problema exterior de Helmholtz:

$$\begin{cases} \nabla^2 u + k^2 u = 0 & \text{en } \mathbb{R}^2 \setminus \Omega, \\ \frac{\partial u}{\partial \mathbf{n}} = g & \text{en } \Gamma, \\ u = f & \text{en } \Gamma, \\ \lim\limits_{r\to\infty} \sqrt{r}\left(\frac{\partial u}{\partial r} - iku\right) = 0. \end{cases} \tag{135}$$

Primeramente se introducen capas de material PML en la dirección x y y para truncar el dominio como se muestran en la Fig. 15. El rectángulo interior contiene el obstáculo Ω y el dominio físico Ω_F.

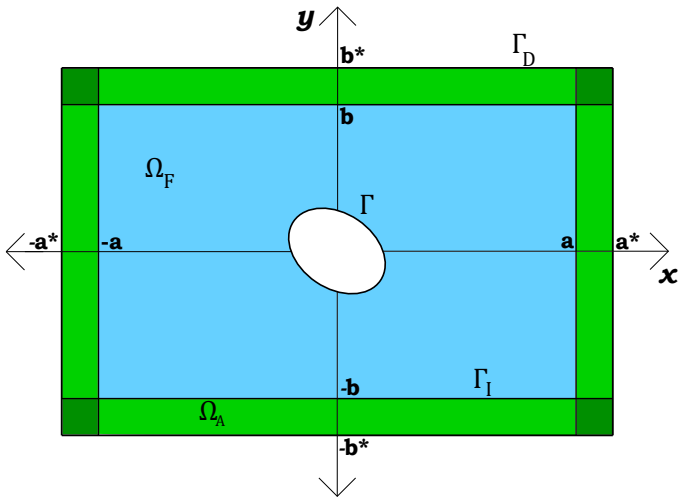

Figura 15: Capa de material PML en un dominio de bidimensional.

La solución aproximada utilizando la capa de PML de la Ec. (135) es

$$\begin{cases} \nabla^2 u + k^2 u = 0 & \text{en } \Omega_F, \\ \frac{1}{\gamma_x}\frac{\partial}{\partial x}\left(\frac{1}{\gamma_x}\frac{\partial u}{\partial x}\right) + \frac{1}{\gamma_y}\frac{\partial}{\partial y}\left(\frac{1}{\gamma_y}\frac{\partial u}{\partial y}\right) + k^2 u = 0 & \text{en } \Omega_A, \\ \frac{\partial u}{\partial \mathbf{n}} = g & \text{en } \Gamma, \\ u = f & \text{en } \Gamma, \\ u \quad \text{y} \quad \left(\frac{1}{\gamma_x}\frac{\partial u}{\partial \nu_x} + \frac{1}{\gamma_y}\frac{\partial u}{\partial \nu_y}\right) & \text{continuo en } \Gamma_I, \\ u = 0 & \text{en } \Gamma_D, \end{cases} \tag{136}$$

donde

$$\gamma_x(x) = \begin{cases} 1, & \text{si } |x| < a, \\ 1 + \frac{i}{\omega}\sigma_x(|x|), & \text{si } a \leq |x| \leq a^*, \end{cases} \tag{137}$$

$$\gamma_y(y) = \begin{cases} 1, & \text{si } |y| < b, \\ 1 + \frac{i}{\omega}\sigma_y(|y|), & \text{si } b \leq |y| \leq b^* \end{cases} \tag{138}$$

y las funciones de absorción son

$$\begin{aligned} \sigma_x(x) &= \tfrac{c}{a^* - x}, \\ \sigma_y(y) &= \tfrac{c}{b^* - y}. \end{aligned} \tag{139}$$

VI.3.1. Problema exterior de Helmholtz 2D

Como vimos en el capítulo anterior se tiene que seguir una serie de pasos para resolver problemas utilizando el FEM:

Generar una partición

Para esto se definirá el dominio $\Omega = \Omega_F \cup \Omega_A$ donde se resolverá el problema, con Ω_A siendo el espacio físico de interés y Ω_F el espacio que contendrá el material PML. El espacio Ω será dividido en elementos triangulares como se muestra en la Fig. 16.

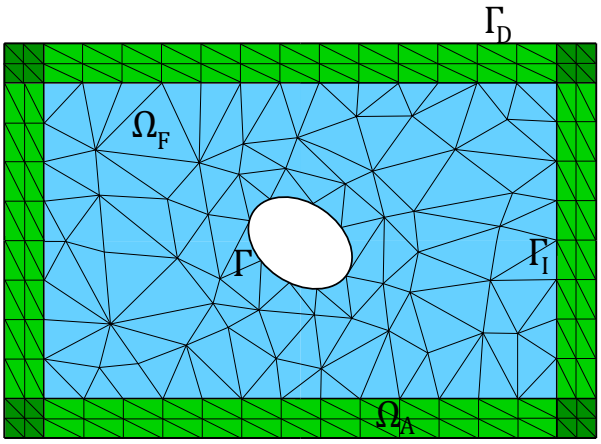

Figura 16: División del dominio físico Ω en elementos triangulares.

Crear el espacio de funciones u_h

Como se trabajará con elementos triangulares se utilizará una aproximación lineal de tres nodos y un grado de libertad por nodo donde cada nodo corresponde a cada vértice del triángulo como se muestra en la Fig. 17.

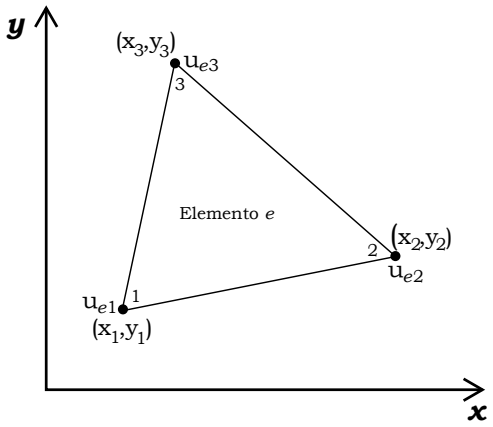

Figura 17: Elemento triangular linear.

Definamos una expresión u_e en términos de u_{e1}, u_{e2} y u_{e3}:

$$u_e = H_1(x,y)u_{e1} + H_2(x,y)u_{e2} + H_3(x,y)u_3, \tag{140}$$

donde

$$H_1 = \frac{1}{2A}\left[(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y\right], \tag{141}$$

$$H_2 = \frac{1}{2A}\left[(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y\right], \tag{142}$$

$$H_3 = \frac{1}{2A}\left[(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y\right] \tag{143}$$

y A_e es el área del elemento e,

$$A_e = \frac{1}{2}det\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}, \tag{144}$$

siendo H_1, H_2 y H_3 las funciones de forma lineales, las cuales satisfacen las siguientes condiciones:

$$H_i(x_j, y_j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \tag{145}$$

y

$$\sum_{i=1}^{3} H_i = 1. \tag{146}$$

Formulación de Galerkin en su forma débil en 2D

En esta sección utilizaremos la formulación de Galerkin en su forma débil aplicada a la Ec. (136),

$$\begin{aligned} &\sum_{e_f} \int_{\Omega_F} w \left(\frac{\partial^2 u}{dx^2} + \frac{\partial^2 u}{dy^2} + k^2 u \right) dxdy + \\ &\sum_{e_a} \int_{\Omega_A} w \left(\frac{1}{\gamma_x} \frac{\partial}{\partial x} \left(\frac{1}{\gamma_x} \frac{\partial u}{\partial x} \right) + \frac{1}{\gamma_y} \frac{\partial}{\partial y} \left(\frac{1}{\gamma_y} \frac{\partial u}{\partial y} \right) + k^2 u \right) dxdy = 0, \end{aligned} \tag{147}$$

donde e_f y e_a son los elementos en la región física de interés y en la región PML respectivamente. Aplicando la primera identidad de Green en la Ec. (147), obtenemos

$$\begin{aligned} &\sum_{e_f} \int_{\Omega_F} \left(\frac{\partial u}{\partial x} \frac{\partial w}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial w}{\partial y} - k^2 uw \right) dxdy + \\ &\sum_{e_a} \int_{\Omega_A} \left(\frac{\gamma_y}{\gamma_x} \frac{\partial u}{\partial x} \frac{\partial w}{\partial x} + \frac{\gamma_x}{\gamma_y} \frac{\partial w}{\partial y} \frac{\partial u}{\partial y} - k^2 \gamma_x \gamma_y uw \right) dxdy = \int_{\Gamma} w \frac{\partial u}{\partial n} ds. \end{aligned} \tag{148}$$

Sustituyendo la Ec. (140) en las integrales de la Ec. (148) y la función de prueba $w_1 = H_1(x, y)$, $w_2 = H_2(x, y)$ y $w_3 = H_3(x, y)$, se tiene que

$$\sum_{e_f} \int_{\Omega_F} \left(\begin{Bmatrix} \frac{\partial H_1}{\partial x} \\ \frac{\partial H_2}{\partial x} \\ \frac{\partial H_3}{\partial x} \end{Bmatrix} \begin{Bmatrix} \frac{\partial H_1}{\partial x} & \frac{\partial H_2}{\partial x} & \frac{\partial H_3}{\partial x} \end{Bmatrix} + \begin{Bmatrix} \frac{\partial H_1}{\partial y} \\ \frac{\partial H_2}{\partial y} \\ \frac{\partial H_3}{\partial y} \end{Bmatrix} \begin{Bmatrix} \frac{\partial H_1}{\partial y} & \frac{\partial H_2}{\partial y} & \frac{\partial H_3}{\partial y} \end{Bmatrix} - k^2 uw \right) dxdy +$$

$$\sum_{e_a}\int_{\Omega_A}\left(\frac{\gamma_y}{\gamma_x}\begin{Bmatrix}\frac{\partial H_1}{\partial x}\\ \frac{\partial H_2}{\partial x}\\ \frac{\partial H_3}{\partial x}\end{Bmatrix}\left\{\frac{\partial H_1}{\partial x}\frac{\partial H_2}{\partial x}\frac{\partial H_3}{\partial x}\right\}+\frac{\gamma_x}{\gamma_y}\begin{Bmatrix}\frac{\partial H_1}{\partial y}\\ \frac{\partial H_2}{\partial y}\\ \frac{\partial H_3}{\partial y}\end{Bmatrix}\left\{\frac{\partial H_1}{\partial y}\frac{\partial H_2}{\partial y}\frac{\partial H_3}{\partial y}\right\}-\right.$$
$$\left.k^2\gamma_x\gamma_y\begin{Bmatrix}H_1\\ H_2\\ H_3\end{Bmatrix}\{H_1H_2H_3\}\right)dxdy=\int_{\Gamma}\begin{Bmatrix}H_1\\ H_2\\ H_3\end{Bmatrix}gds. \tag{149}$$

La Ec. (149) contiene las dos regiones que componen el domino Ω, la región que representa el dominio físico o de interés Ω_F y la región del material PML en Ω_A.

Solución del problema

Igual que en el problema en 1D se genera un sistema de ecuaciones lineales, el cual puede ser resuelto utilizando los métodos antes mencionados.

VI.3.2. Problemas en 2D

En esta sección se utilizarán los resultados obtenidos en la sección anterior para resolver dos problemas que involucran una fuente emitiendo en el espacio. En el primero tendrá un medio uniforme y en el segundo estará formado por dos medios, los cuales tendrán índices de refracción distintos y esto nos ayudará a observar la *Ley de Snell*.

Problema 1:
Índice de refracción continuo

Para contrastar un campo conocido con el obtenido por el FEM combinado con la técnica PML, vamos a considerar que en nuestra región de estudio se tiene una fuente circular que está emitiendo. El campo generado por la fuente estará dado por la solución fundamental

$$\Phi(x,y)=\frac{i}{4}H_0^{(1)}(k|x-y|), \tag{150}$$

donde $H_0^{(1)}$ es la función de Hankel de primera especie y orden cero.

Vamos a contrastar el campo teórico con el obtenido por el FEM combinado con la técnica PML; conociendo sólo el valor en la frontera interior Γ la cual funcionará como una fuente.

Utilizando el programa *Gmsh* se hará un mallado del dominio Ω, el cual contará con 6258 nodos y 12220 triángulos como se muestra en Fig. 18.

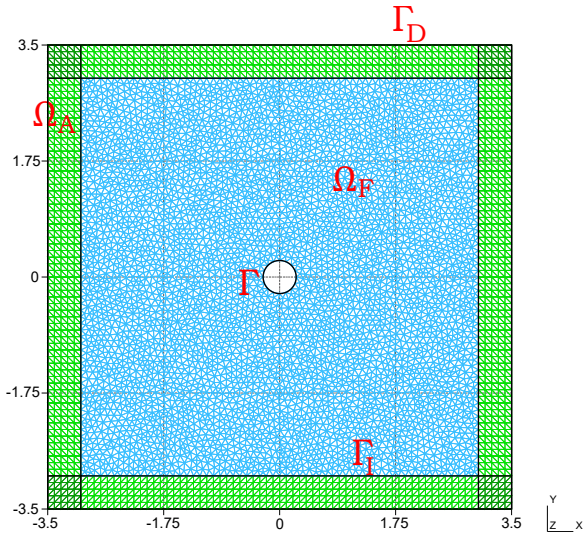

Figura 18: Mallado para el problema 1 con 6258 nodos y 12220 triángulos.

Para el cálculo del campo, el problema puede ser expresado como

$$\begin{cases} \nabla^2 u + k^2 u = 0 & \text{en } \Omega_F, \\ \frac{1}{\gamma_x}\frac{\partial}{\partial x}\left(\frac{1}{\gamma_x}\frac{\partial u}{\partial x}\right) + \frac{1}{\gamma_y}\frac{\partial}{\partial y}\left(\frac{1}{\gamma_y}\frac{\partial u}{\partial y}\right) + k^2 u = 0 & \text{en } \Omega_A, \\ u = \Phi(x,y) & \text{en } \Gamma, \\ u \quad \text{y} \quad \left(\frac{1}{\partial\gamma_x}\frac{\partial u}{\partial \nu_x} + \frac{1}{\partial\gamma_y}\frac{\partial u}{\partial \nu_y}\right) & \text{continuo en } \Gamma_I, \\ u = 0 & \text{en } \Gamma_D. \end{cases} \tag{151}$$

Adaptando la Ec. (149) a la Ec. (151) obtenemos:

$$\sum_{e_f}\int_{\Omega_F}\left(\begin{Bmatrix}\frac{\partial H_1}{\partial x}\\ \frac{\partial H_2}{\partial x}\\ \frac{\partial H_3}{\partial x}\end{Bmatrix}\begin{Bmatrix}\frac{\partial H_1}{\partial x}\frac{\partial H_2}{\partial x}\frac{\partial H_3}{\partial x}\end{Bmatrix} + \begin{Bmatrix}\frac{\partial H_1}{\partial y}\\ \frac{\partial H_2}{\partial y}\\ \frac{\partial H_3}{\partial y}\end{Bmatrix}\begin{Bmatrix}\frac{\partial H_1}{\partial y}\frac{\partial H_2}{\partial y}\frac{\partial H_3}{\partial y}\end{Bmatrix} - k^2 uw\right)dxdy +$$

$$\sum_{e_a}\int_{\Omega_A}\left(\frac{\gamma_y}{\gamma_x}\begin{Bmatrix}\frac{\partial H_1}{\partial x}\\ \frac{\partial H_2}{\partial x}\\ \frac{\partial H_3}{\partial x}\end{Bmatrix}\left\{\frac{\partial H_1}{\partial x}\frac{\partial H_2}{\partial x}\frac{\partial H_3}{\partial x}\right\}+\frac{\gamma_x}{\gamma_y}\begin{Bmatrix}\frac{\partial H_1}{\partial y}\\ \frac{\partial H_2}{\partial y}\\ \frac{\partial H_3}{\partial y}\end{Bmatrix}\left\{\frac{\partial H_1}{\partial y}\frac{\partial H_2}{\partial y}\frac{\partial H_3}{\partial y}\right\}-\right.$$
$$\left.k^2\gamma_x\gamma_y\begin{Bmatrix}H_1\\ H_2\\ H_3\end{Bmatrix}\{H_1H_2H_3\}\right)dxdy=0. \tag{152}$$

Dicha ecuación se utilizará para realizar el modelado numérico del problema.

Programa 6

En esta sección se mostrará el programa que utiliza el FEM en combinación la técnica PML para resolver la Ecuación de Helmholtz 2D aplicado al *Problema 1.*

****

Programa 6. Método de Elementos Finitos en combinación la técnica PML para la Ecuación de Helmholtz 2D aplicado al Problema 1.

****

```
function [U,Ui,p,t]=Helmholtz2D3P_PML_Problema1(p,t,front)
     % p=msh.POS;%puntos
     % t=msh.TRIANGLES;%triangulos
     % front=msh.LINES;%frontera

     r=0.25; % Radio de la fuente
     disx=3; % a
     disy=3; % b
     pmlx=0.5; % Tamaño de la regi\'on PML en el eje x
     pmly=0.5; % Tamaño de la regi\'on PML en el eje y
     dpmlx=pmlx+0.0001; % Suma 0.0001 para evitar divici\'on entre cero
     dpmly=pmly+0.0001; % Suma 0.0001 para evitar divici\'on entre cero
%---------------------------------------------------------------------
     k0=4; % Constante k
     d=0; % Constante que mueve la fuente en el eje x
     a=complex(0,1); % N\'umero complejo 0 + i
     h=@(x,y)r.*a.*(besselj(0,k0.*sqrt((x+d).^2+y.^2))+a.*...
        bessely(0,k0.*sqrt((x+d).^2+y.^2))); % función Psi(x,y)
%---------------------------------------------------------------------
```

```
     t(:,4)=[]; % Eliminaci\'on columna 4 de t "no relevante"
     p(:,3)=[]; % Eliminaci\'on columna 3 de p "no relevante"
     front=front(:,1); % Utilizaci\'on de la columna 1 de front por
                       % simplicidad
     frontera=[]; % Inicializac\'on de la lista frontera

     N=size(p,1); % N\'umero de puntos
     T=size(t,1); % N\'umero de triangulos
     K=sparse(N,N); % Inicializacion Matriz de rigideces
     M=sparse(N,N); % Inicializacion Matriz de masas
     F=zeros(N,1); % Inicialización Vector de cargas

     tem=sparse(N,1); % Vector que almacer\a la interaci\'on de la
                      % frontera con la matriz global
     tem1=ones(N,1); % Vector que almacenar\'a la ubicaci\'on de
                     % los nodos interiores
     tem2=zeros(N,1); % Vector para forzar condici\'on de frontera

     Ke=zeros(3,3); % Matriz auxiliar Ke
     Me=zeros(3,3); % Matriz auxiliar Me

%--------------------------------------------------------------------
     % Calculo de las matrices auxiliares Ke, Me
     % y ensamblaje de la matriz de riguidez K y de masas M
     for e=1:T % Integraci\'on sobre los elementos
         [t,x,y,J,nodes]=Correccion_orientacion(t,e,p); % Correcci\'on
                       % de orientaci\'on de los Tri\'angulos
          val=Revisa_Ubicacion(x,y,disx,disy); % Revisa la ubicaci\'on
                       % del elemento retorna 1 si se encuentra en la
                       % regui\'on de estudio y 2 en la regi\'on PML
          if(val==1) % Regi\'on de estudio
               [Ke,Me]=Region_Estudio(x,y,J,k0); % Calculo de la  ma-
                       % trices Ke y Me para la regi\'on de estudio
          else % Regi\'on PML
               [Ke,Me]=Region_PML(x,y,J,k0,dpmlx,dpmly,disx,disy);
               % Calculo de la matrices Ke y Me para la regi\'on PML
          end % Fin if
          % Ensambaje de la matriz de riguideces K y de masas M
          for i=1:3
               for j=1:3
                    K(nodes(i),nodes(j))=K(nodes(i),nodes(j))+Ke(i,j);
                    M(nodes(i),nodes(j))=M(nodes(i),nodes(j))+Me(i,j);
               end % Fin for j
          end % Fin for i
```

```
     end % Fin for e
%-------------------------------------------------------------------------
     % Ahora de a\~naden las condiciones de Dirichlet
     pf=p(front(),:); % Puntos frontera
     for i=1:length(front) % Ciclo para agregar condici\'n de Dirichlet
          px=pf(i,1); % Punto de la frontera x
          py=pf(i,2); % Punto de la frontera y
          cir=px^2+py^2; % Distancia del origen al punto frontera

          % Nota: front contiene los puntos en Gamma, Gamma_{I} y
          % Gamma_{D}
          % If que solo admite puntos en Gamma y Gamma_{D}
          if(px==(disx+pmlx)||px==-(disx+pmlx)||py==(disy+pmly)||...
                          py==-(disy+pmly)||abs(cir-(r)^2)<=0.00001)
               frontera=[frontera,front(i)]; % Almacena los puntos de
                          % Gamma y Gamma_{D}
               if(abs(cir-(r)^2)<=0.00001) % If que solo admite puntos
                          % en Gamma para la condici\'on Dirichlet
                    tem=tem+h(px,py).*K(front(i),:)'.*ones(N,1)+...
                         h(px,py).*M(front(i),:)'.*ones(N,1);% Vec-
                          % tor que almacena la interaci\'on de la
                          % condici\'on de frontera con la  matriz
                          % Global K y M
                    tem2(front(i),1)=h(px,py); % Vector para forzar
                          % condici\'on de Dirichlet en el vector
                          % de cargas F
               end % Fin if
          end % Fin if
     end % Fin for
  %-------------------------------------------------------------------------
     tem1(frontera)=0; % Elementos internos con "1","0" los nodos fron-
                       %tera
     F=F-tem; % Suma del vector de interaci\'on de la condici\'n de la
              % frontera con la matriz Global K al vector de cargas
     F=tem1.*F+tem2; % Forzar condici\'on de frontera en el vector de
                     % cargas
%-------------------------------------------------------------------------
     K(frontera,:)=0; K(:,frontera)=0; % Ponemos ceros para no modificar
                                       % los  valores en la frontera
     K(frontera,frontera)=speye(length(frontera),length(frontera)); % Se
                            % pone 1/2  en la submatriz "frontera" de K
     M(frontera,:)=0; M(:,frontera)=0; % Ponemos ceros para no modificar
                              % los  valores en la frontera
     M(frontera,frontera)=speye(length(frontera),length(frontera)); % Se
                            % pone 1/2  en la submatriz "frontera" de M
```

```
     % Nota: El 1/2 en cada punto de la frontera es para que en el mo-
     % mento de sumar K+M la nueva matriz tenga 1 en los puntos frontera
%-------------------------------------------------------------------------
     % Ahora resolvemos
     Ui=h(p(:,1),p(:,2)); % Soluci\'on Anal\'itica
     U=(K+M)\F; % La aproximaci\'on es U_1 phi_1 + ... + U_N phi_N
end % Fin del programa
```

Los algoritmos utilizadas por el programa 6 se pueden encontrar en el apéndice E.

En la Fig. 19 se puede observar la solución numérica y analítica del campo del problema 1.

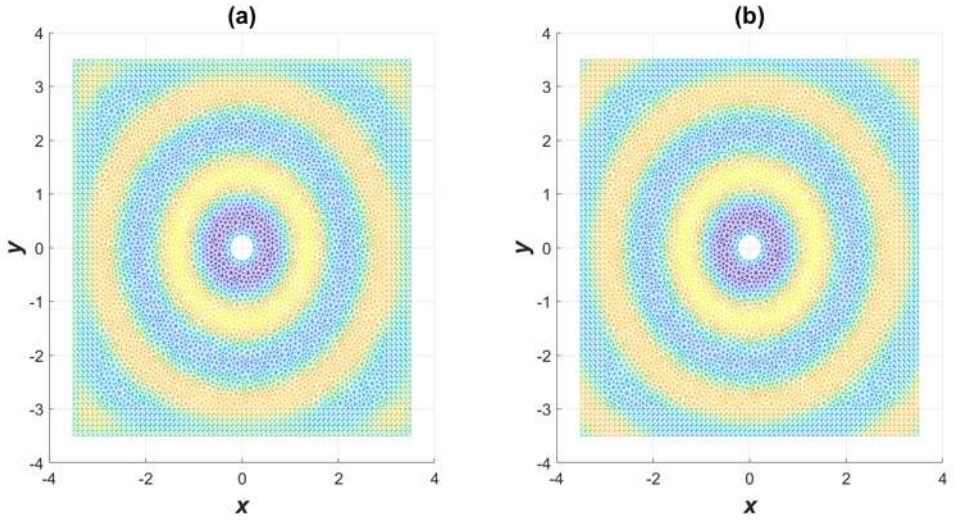

Figura 19: Solución al problema 1. a) Parte real del campo calculado mediante FEM. b) Parte real del campo teórico (mediante función de Hankel).

Problema 2:
Dos Medios

El problema de dos medios es muy similar al anterior, salvo que nuestra región de interés estará dividida en dos regiones, donde cada uno tendrá un índice de refracción distinto. Esto con el fin de observar la *Ley de Snell*.

Para el cálculo del campo, el problema puede ser expresado como

$$\begin{cases} \nabla^2 u + k_1^2 u = 0 & \text{en } \Omega_{F_1}, \\ \nabla^2 u + k_2^2 u = 0 & \text{en } \Omega_{F_2}, \\ \frac{1}{\gamma_x}\frac{\partial}{\partial x}\left(\frac{1}{\gamma_x}\frac{\partial u}{\partial x}\right) + \frac{1}{\gamma_y}\frac{\partial}{\partial y}\left(\frac{1}{\gamma_y}\frac{\partial u}{\partial y}\right) + k_1^2 u = 0 & \text{en } \Omega_{A_1}, \\ \frac{1}{\gamma_x}\frac{\partial}{\partial x}\left(\frac{1}{\gamma_x}\frac{\partial u}{\partial x}\right) + \frac{1}{\gamma_y}\frac{\partial}{\partial y}\left(\frac{1}{\gamma_y}\frac{\partial u}{\partial y}\right) + k_2^2 u = 0 & \text{en } \Omega_{A_2}, \\ u = \Phi(x,y) & \text{en } \Gamma, \\ u \quad \text{y} \quad \frac{\partial u}{\partial \mathbf{n}} & \text{continuo en } \Gamma_F, \\ u \quad \text{y} \quad \left(\frac{1}{\partial\gamma_x}\frac{\partial u}{\partial \nu_x} + \frac{1}{\partial\gamma_y}\frac{\partial u}{\partial \nu_y}\right) & \text{continuo en } \Gamma_I, \\ u = 0 & \text{en } \Gamma_D, \end{cases} \tag{153}$$

donde $k_1 = n_1\omega_r$ y $k_2 = n_2\omega_r$ con $n_1 = 1$ y $n_2 = 1.333$. Tras calcular la solución representamos su parte real como se muestra en la Fig. 20.

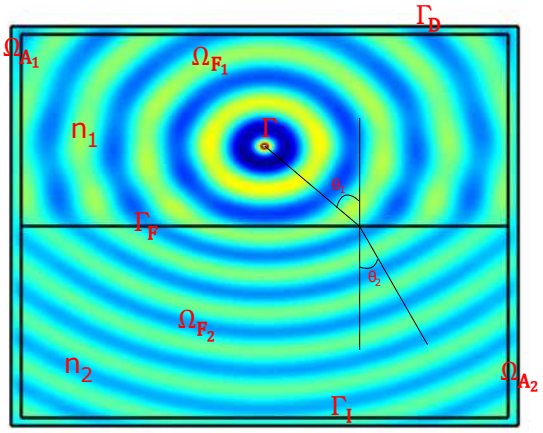

Figura 20: Solución al problema 2. Ley de snell con $\theta_1 = 50°$ y $\theta_2 = 35°$.

En la Fig. 20 se puede observar que al cambiar de un medio a otro el frente de onda cambia su ángulo de propagación de acuerdo con la *Ley de snell* así como se genera interferencia por las ondas reflejadas.

Programa 7

En esta sección se mostrará el programa utilizado para resolver el *Problema 2*.

****

Programa 7. Método de Elementos Finitos en combinación del Método PML para la Ecuación de Helmholtz 2D aplicado al Problema 2.

****

```
function [U,p,t,frontera,circulo]=Helmholtz2D3P_PML_Problema2(p,t,front)
     % p=msh.POS;%puntos
     % t=msh.TRIANGLES;%triangulos
     % front=msh.LINES;%frontera

     r=0.25; % Radio de la fuente
     disx=24; % a
     disy=24; % b
     pmlx=1; % Tamaño de la regi\'on PML en el eje x
     pmly=1; % Tamaño de la regi\'on PML en el eje y
     dpmlx=pmlx+0.0001; % Suma 0.0001 para evitar divici\'on entre cero
     dpmly=pmly+0.0001; % Suma 0.0001 para evitar divici\'on entre cero
%---------------------------------------------------------------------
     d=10; % Constante que mueve la fuente en el eje x
     a=complex(0,1); % N\'umero complejo 0 + i
     h=@(x,y)r.*a.*(besselj(0,k0.*sqrt((x+d).^2+y.^2))+a.*...
     bessely(0,k0.*sqrt((x+d).^2+y.^2))); % función Psi(x,y)
%---------------------------------------------------------------------
     t(:,4)=[]; % Eliminaci\'on columna 4 de t "no relevante"
     p(:,3)=[]; % Eliminaci\'on columna 3 de p "no relevante"
     front=front(:,1); % Utilizaci\'on de la columna 1 de front por
                       % simplicidad
     frontera=[]; % Inicializac\'on de la lista frontera
     circulo=[]; % Inicializac\'on de la lista circulo

     N=size(p,1); % N\'umero de puntos
     T=size(t,1); % N\'umero de triangulos
     K=sparse(N,N); % Inicializacion Matriz de rigideces
     M=sparse(N,N); % Inicializacion Matriz de masas
     F=zeros(N,1); % Inicialización Vector de cargas
     tem=sparse(N,1); % Vector que almacer\a la interaci\'on de la
                      % frontera con la matriz global
     tem1=ones(N,1); % Vector que almacenar\'a la ubicaci\'on de
                     % los nodos interiores
```

```
     tem2=zeros(N,1); % Vector para forzar condici\'on de frontera

     Ke=zeros(3,3); % Matriz auxiliar Ke
     Me=zeros(3,3); % Matriz auxiliar Me
%--------------------------------------------------------------------
     % Calculo de las matrices auxiliares Ke, Me
     % y ensamblaje de la matriz de riguidez K y de masas M
     for e=1:T % Integraci\'on sobre los elementos
          [t,x,y,J,nodes]=Correccion_orientacion(t,e,p); % Correcci\'on
                    % de orientaci\'on de los Tri\'angulos
          val=Revisa_Ubicacion(x,y,disx,disy); % Revisa la ubicaci\'on
                    % del elemento retorna 1 si se encuentra en la
                    % regui\'on de estudio y 2 en la regi\'on PML
          if(x(1)<=0 && x(2)<=0 && x(3)<=0) % Regi\'on omega_{F_{1}}
               k0=1.0; % n_-{1}=1.0
          else % Regi\'on omega_{F_{2}}
               k0=1.333; % n_{2}=1.333
          end % Fin del if
          if(val==1) % Regi\'on de estudio
               [Ke,Me]=Region_Estudio(x,y,J,k0); % Calculo de la  ma-
                         % trices Ke y Me para la regi\'on de estudio
          else % Regi\'on PML
               [Ke,Me]=Region_PML(x,y,J,k0,dpmlx,dpmly,disx,disy);
               % Calculo de la matrices Ke y Me para la regi\'on PML
          end % Fin if
          % Ensambaje de la matriz de riguideces K y de masas M
          for i=1:3
               for j=1:3
                    K(nodes(i),nodes(j))=K(nodes(i),nodes(j))+Ke(i,j);
                    M(nodes(i),nodes(j))=M(nodes(i),nodes(j))+Me(i,j);
               end % Fin for j
          end % Fin for i
     end % Fin for e
%--------------------------------------------------------------------
     % Ahora de a\~naden las condiciones de Dirichlet
     pf=p(front(),:);  % Puntos frontera
     k0=1.0; % Fuente en la regi\'on 1 por tanto n_{1}=1.0
     for i=1:length(front) % Ciclo para agregar condici\'n de Dirichlet
          px=pf(i,1); % Punto de la frontera x
          py=pf(i,2); % Punto de la frontera y
          cir=(px+d)^2+py^2; % Distancia del origen al punto frontera
          % Nota: front contiene los puntos en Gamma, Gamma_{I},
          % Gamma_{D} y Gamma_{F}
          % If que solo admite puntos en Gamma y Gamma_{D}
          if(abs(px)==(disx+pmlx)||abs(py)==(disy+pmly)||...
```

```
                                abs(cir-(r)^2)<=0.00001)
               frontera=[frontera,front(i)]; % Almacena los puntos de
               % Gamma y Gamma_{D}
               if(abs(cir-(r)^2)<=0.00001) % If que solo admite puntos
                           % en Gamma para la condici\'on Dirichlette
                    tem=tem+h(px,py).*K(front(i),:)'.*ones(N,1)+...
                    h(px,py).*M(front(i),:)'.*ones(N,1); % Vec-
                              % tor que almacena la interaci\'on de la
                              % condici\'on de frontera con la  matriz
                              % Global K y M
                    tem2(front(i),1)=h(px,py); % Vector para forzar
                            % condici\'on de Dirichlet en el vector
                            % de cargas F
                    circulo=[circulo,front(i)]; % Lista que almacena
                                        % los puntos que se encuen-
                                        % tran en la fuente
               end % Fin if
          end % Fin if
     end % Fin for i
%-------------------------------------------------------------------------
     tem1(frontera)=0; % Elementos internos con "1","0" los nodos fron-
                       % tera
     F=F-tem; % Suma del vector de interaci\'on de la condici\'n de la
              % frontera con la matriz Global K al vector de cargas
     F=tem1.*F+tem2; % Forzar condici\'on de frontera en el vector de
                     % cargas
     K(frontera,:)=0; K(:,frontera)=0; % Ponemos ceros para no modi-
                                    % ficar los valores en la frontera
     K(frontera,frontera)=speye(length(frontera),length(frontera))/2;
                       % se pone 1/2  en la submatriz "frontera" de K
     M(frontera,:)=0; M(:,frontera)=0; % Ponemos ceros para no modi-
                                     % ficarlos valores en la frontera
     M(frontera,frontera)=speye(length(frontera),length(frontera))/2;
                      % se pone 1/2  en la submatriz "frontera" de M
     % Nota: El 1/2 en cada punto de la frontera es para que en el mo-
     %       mento de sumar K+M la nueva matriz tenga 1 en los puntos
     %       frontera
%-------------------------------------------------------------------------
     % Ahora resolvemos
     U=(K+M)\F; % La aproximaci\'on es U_1 phi_1 + ... + U_N phi_N
     circulo=[circulo,circulo(1)]; % Se agrega el primer punto al
     % final para cerrar al momento de graficar
end % Fin del programa
```

Los algoritmos utilizadas por el programa 7 se pueden encontrar en el apéndice E.

VI.4. Guía de ondas de conductor perfecto

En esta sección se aplicará los resultados obtenidos a lo largo del trabajo al campo de la óptica, y más concretamente a guías de ondas de conductor perfecto (ver Fig. 21). Se considerará una región en dos dimensiones y se observará como el campo es afectado por la existencia de obstáculos en el interior de la guía de ondas. El campo estará gobernado por la ecuación de Helmholtz en la que el número de onda, k, se tomará como un número real positivo en función de la frecuencia reducida (ω_r).

Se utilizará la técnica de descomponer el campo u en incidente u^i y esparcido u^s de forma que

$$u = u^s + u^i, \tag{154}$$

donde el campo u^i será una fuente.

La geometría del sistema que se consideró se muestra en la Fig. 21. Un sistema formado por dos placas paralelas Ω_1, Ω_2 y cinco inclusiones circulares $\Omega_3, ..., \Omega_7$ de conductor perfecto, el cual será iluminado por un haz Gaussiano. Por simplicidad en este trabajo únicamente se presenta el caso de polarización TE; es decir

$$u = 0 \qquad \text{en} \qquad \Omega_1, \Omega_2, ..., \Omega_7 \tag{155}$$

y el campo en sus fronteras $\Gamma_1, \Gamma_2, ..., \Gamma_7$ será nulo. Por ello $u^s = -u^i$.

Para el cálculo del campo, el problema puede ser expresado como

$$\begin{cases} \nabla^2 u^s + k^2 u^s = 0 & \text{en } \Omega_F, \\ \frac{1}{\gamma_x}\frac{\partial}{\partial x}\left(\frac{1}{\gamma_x}\frac{\partial u^s}{\partial x}\right) + \frac{1}{\gamma_y}\frac{\partial}{\partial y}\left(\frac{1}{\gamma_y}\frac{\partial u^s}{\partial y}\right) + k^2 u^s = 0 & \text{en } \Omega_A, \\ u^s = -u^i & \text{en } \Gamma_1, \Gamma_2, ..., \Gamma_7, \\ u^s \quad \text{y} \quad \left(\frac{1}{\partial\gamma_x}\frac{\partial u^s}{\partial \nu_x} + \frac{1}{\partial\gamma_y}\frac{\partial u^s}{\partial \nu_y}\right) & \text{continuo en } \Gamma_I, \\ u^s = 0 & \text{en } \Gamma_D. \end{cases} \tag{156}$$

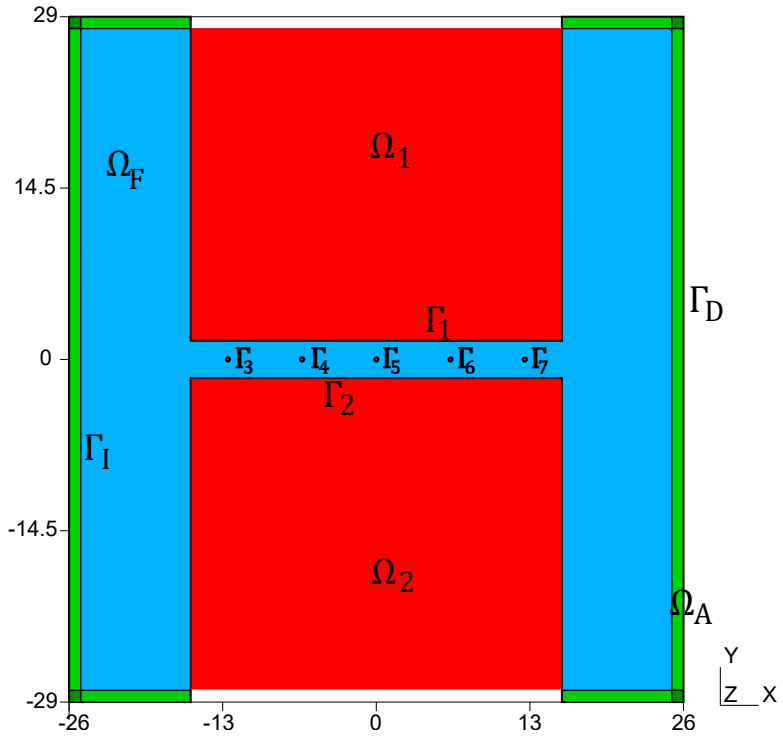

Figura 21: Perfil de la guía de ondas con 5 inclusiones circulares. La región Ω_1 y Ω_2 constituyen las placas paralelas conductoras, mientras que las regiones Ω_3, Ω_4, Ω_5, Ω_6 y Ω_7 constituyen las inclusiones circulares conductoras.

Programa 8

En esta sección se exhibe la implementación del programa que utiliza el FEM en combinación con el Método PML para calcular el campo de una PCW de conductor perfecto.

****

Programa 8. Método de Elementos Finitos en combinación del Método PML para calcular el campo de una PCW.

****

```
function [U,Us,Ui,p,t,frontera]=Helmholtz2D3P_PML_PCW(p,t,front)
     % p=msh.POS; % Puntos
     % t=msh.TRIANGLES; % Tri\'angulos
     % front=msh.LINES; % Frontera

     Inclusiones=5; % N\'umero de inclusiones
```

```
     pii=3.1416; % Constante pi
     ll=pii*Inclusiones; % Longitud de la gu\'ia de ondas
     hh=1.5708; % Ancho
     d=ll; % Longitud de la gu\'ia de ondas d
     r=0.1772; % Radio de las incluiones
     disx=25; % Anclo de la regi\'on de estudio Omega_{F}
     disy=28; % Alto de la regi\'on de estudio Omega_{F}
     pmlx=1; % Espesor de la regi\'on PML en el eje x
     pmly=1; % Espesor de la regi\'on OML en el eje y
     dpmlx=pmlx+0.0001; % Suma 0.0001 para evitar divici\'on entre cero
     dpmly=pmly+0.0001; % Suma 0.0001 para evitar divici\'on entre cero
%--------------------------------------------------------------------------
     k0=3.0; % Constante k=n*w_{r}, n=1, w_{r}={1.3843843937,1.50000,
                                                % 2.0,2.1371371746,3.0}
     theta=0.0; % \'Angulo de incidencia del haz Gaussiano
     nwa=28; % Constante para el haz Gausiano
     wnum=k0; % Frecuencia reducidad
     g=2.0*nwa/5.0; % Constante para el haz Gausiano
%--------------------------------------------------------------------------
     t(:,4)=[]; % Eliminaci\'on columna 4 de t "no relevante"
     p(:,3)=[];  % Eliminaci\'on columna 3 de p "no relevante"
     front=front(:,1); % Utilizaci\'on de la columna 1 de front por
                       % simplicidad
     frontera=[]; % Inicializac\'on de la lista frontera

     N=size(p,1); % N\'umero de puntos
     T=size(t,1); % N\'umero de triangulos
     K=sparse(N,N); % Inicializacion Matriz de rigideces
     M=sparse(N,N); % Inicializacion Matriz de masas
     F=zeros(N,1); % Inicialización Vector de cargas
     tem=sparse(N,1); % Vector que almacer\a la interaci\'on de la
                      % frontera con la matriz global
     tem1=ones(N,1); % Vector que almacenar\'a la ubicaci\'on de
                     % los nodos interiores
     tem2=zeros(N,1); % Vector para forzar condici\'on de frontera
     Ke=zeros(3,3); % Matriz auxiliar Ke
     Me=zeros(3,3); % Matriz auxiliar Me
%--------------------------------------------------------------------------
     % Calculo de las matrices auxiliares Ke, Me y ensamblaje de la
     % matriz de riguidez K y de masas M
     for e=1:T % Integraci\'on sobre los elementos
          [t,x,y,J,nodes]=Correccion_orientacion(t,e,p); % Correc-
                     % ci\'on de orientaci\'on de los Tri\'angulos
          val=Revisa_Ubicacion(x,y,disx,disy); % Revisa la ubica-
                 % ci\'on del elemento retorna 1 si se encuentra en
```

```
                % la regui\'on de estudio y 2 en la regi\'on PML
          if(val==1) % Regi\'on de estudio
               [Ke,Me]=Region_Estudio(x,y,J,k0); % Calculo de la
                       % matrices Ke y Me para la regi\'on de estudio
          else % Regi\'on PML
              [Ke,Me]=Region_PML(x,y,J,k0,dpmlx,dpmly,disx,disy);
              % Calculo de la matrices Ke y Me para la regi\'on PML
          end % Fin if
     % Ensambaje de la matriz de riguideces K y de masas M
          for i=1:3
               for j=1:3
                    K(nodes(i),nodes(j))=K(nodes(i),nodes(j))+Ke(i,j);
                    M(nodes(i),nodes(j))=M(nodes(i),nodes(j))+Me(i,j);
               end % Fin for j
          end % Fin for i
     end % Fin for e
%-------------------------------------------------------------------------
     % Ahora de a\~naden las condiciones de Dirichlet
     pf=p(front(),:); % Puntos frontera
     for i=1:length(front) % Ciclo para agregar condici\'n de Dirichlet
          px=pf(i,1); % Punto de la frontera x
          py=pf(i,2); % Punto de la frontera y
          cir1=(px+4*pii)^2+py^2; % Inclusi\'on 1
          cir2=(px+2*pii)^2+py^2; % Inclusi\'on 2
          cir3=(px+0*pii)^2+py^2; % Inclusi\'on 3
          cir4=(px-2*pii)^2+py^2; % Inclusi\'on 4
          cir5=(px-4*pii)^2+py^2; % Inclusi\'on 5
          % Nota: front contiene los puntos en Gamma, Gamma_{I},
          % Gamma_{D}, Gamma_{F}, Gamma_{1}, Gamma_{2}, Gamma_{3},
          % Gamma_{4}, Gamma_{5}
          % If que solo admite puntos en Gamma, Gamma_{1}, Gamma_{2},
          % Gamma_{3}, Gamma_{4}, Gamma_{5} y Gamma_{D}
          if(abs(cir1-(r)^2)<=0.00001 || abs(cir2-(r)^2)<=0.00001...
             || abs(cir3-(r)^2)<=0.00001 || abs(cir4-(r)^2)<=0.00001...
             || abs(cir5-(r)^2)<=0.00001|| abs(py)==(disy+pmly)...
             ||abs(px)==(disx+pmlx) || abs(px)==ll || (abs(py)==hh ...
             && abs(px)<=ll))
               frontera=[frontera,front(i)]; % Almacena los puntos de
                              % Gamma, Gamma_{1}, Gamma_{2}, Gamma_{3},
                              % Gamma_{4}, Gamma_{5} y Gamma_{D}
               if(abs(cir1-(0.1772)^2)<=0.00001 ...
                    || abs(cir2-(0.1772)^2)<=0.00001 ...
                    || abs(cir3-(0.1772)^2)<=0.00001 ...
                    || abs(cir4-(0.1772)^2)<=0.00001 ...
                    || abs(cir5-(0.1772)^2)<=0.00001...
```

```
                    || abs(px)==ll || (abs(py)==hh && abs(px)<=ll))
                    % If que solo admite puntos en Gamma, Gamma_{1},
                    % Gamma_{2}, Gamma_{3}, Gamma_{4}, Gamma_{5} para
                    % la condici\'on Dirichlette
                    tem=tem-haz_gaussiano(px,py,wnum,g,theta,d)...
                    .*K(front(i),:)'.*ones(N,1)...
                    -haz_gaussiano(px,py,wnum,g,theta,d)...
                    .*M(front(i),:)'...
                    .*ones(N,1); % Vector que almacena la interaci\'on
                                 % de la condici\'on de frontera con la
                                 % matriz Global K y M
                    tem2(front(i),1)=-haz_gaussiano(px,py,wnum,g,...
                         theta,d);
                         % Vector para forzar condici\'on de Dirichlet
                         % en el vector de cargas F
               end % Fin if
          end % Fin if
     end % Fin for i
%-------------------------------------------------------------------------
     tem1(frontera)=0; % Elementos internos con "1","0" los nodos fron-
                       % tera
     F=F-tem; % Suma del vector de interaci\'on de la condici\'n de la
              % frontera con la matriz Global K al vector de cargas
     F=tem1.*F+tem2; % Forzar condici\'on de frontera en el vector de
                     % cargas
     K(frontera,:)=0; K(:,frontera)=0; % Ponemos ceros para no modificar
                                       % los valores en la frontera
     K(frontera,frontera)=speye(length(frontera),length(frontera))/2;
                       % se pone 1/2  en la submatriz "frontera" de K
     M(frontera,:)=0; M(:,frontera)=0; % Ponemos ceros para no modificar
                                       % los valores en la frontera
     M(frontera,frontera)=speye(length(frontera),length(frontera))/2;
                       % se pone 1/2  en la submatriz "frontera" de M
     % Nota: El 1/2 en cada punto de la frontera es para que en el mo-
     % mento de sumar K+M la nueva matriz tenga 1 en los puntos frontera
%-------------------------------------------------------------------------
     % Ahora resolvemos
     Ui=haz_gaussiano(p(:,1),p(:,2),wnum,g,theta,d); % Campo incidente
                                                     % u^i
     Us=(K+M)\F; % Campo esparcido u^s
     U=Us+Ui; % Campo total
end % Fin del programa
```

Los algoritmos utilizadas por el programa 8 se pueden encontrar en el apéndice E.

Capítulo VII

CÁLCULO DE LA INTENSIDAD DEL CAMPO DE UNA PCW USANDO IEM Y FEM

En este capítulo se mostrarán los resultados numéricos obtenidos al calcular la intensidad del campo cercano de una PCW de conductor perfecto utilizando el Método de la Ecuación Integral y el Método de Elementos Finitos, los cuales fueron explicados en el capítulo III y V respectivamente.

VII.1. Campo cercano de una PCW de material conductor perfecto

En esta parte del trabajo se mostrará los resultados obtenidos al calcular la intensidad del campo cercano de una PCW de conductor perfecto utilizando IEM y FEM. Para esto se considera una PCW con una longitud $d = 10\pi$, un ancho $b = \pi$ y con cinco inclusiones circulares de radio $r = 0.177$ (ver Fig. 21).

VII.1.1. Intensidad del campo esparcido de una PCW de conductor perfecto

En las Figs. 22 y 23 se muestran los resultados obtenidos de las intensidades de los campos cercano esparcidos calculados con el IEM y FEM para las frecuencias $\omega_r = 1.3843$, $\omega_r = 1.5$, $\omega_r = 2.1371$ y $\omega_r = 3.0$.

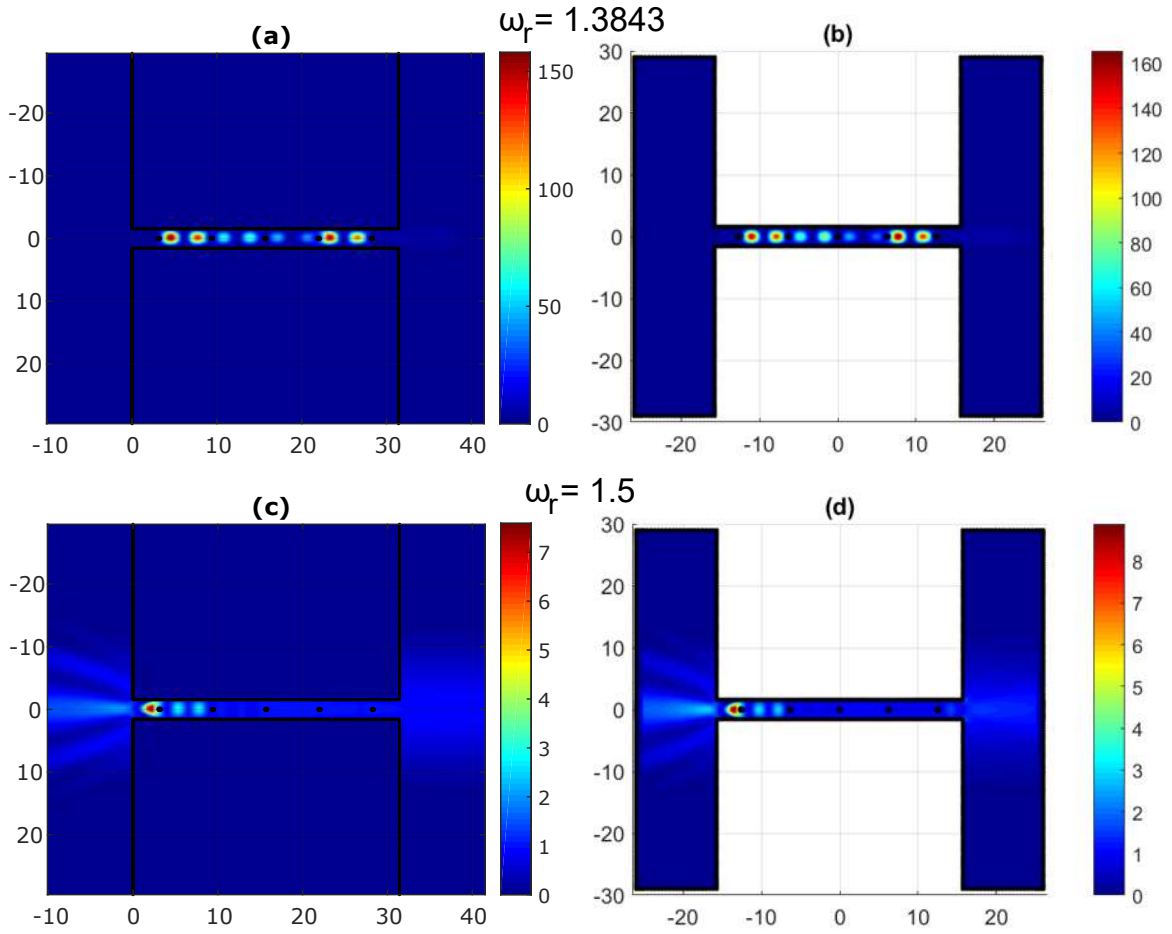

Figura 22: Intensidades del campo cercano esparcido de una PCW de conductor perfecto mediante ((a) y (c)) IEM y ((b) y (d)) FEM.

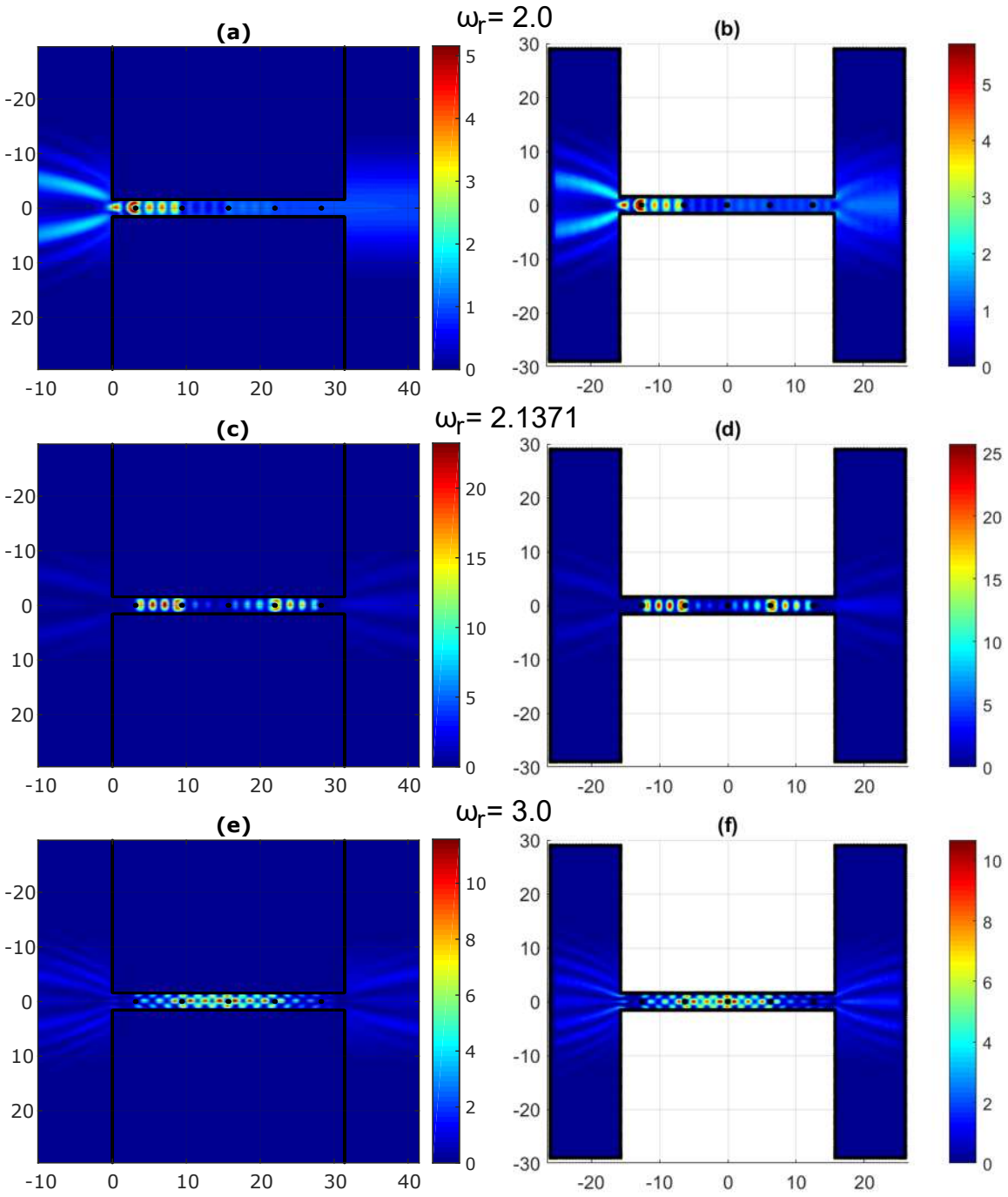

Figura 23: Intensidades del campo cercano esparcido de una PCW de conductor perfecto mediante ((a), (c) y (e)) IEM y ((b), (d) y (f)) FEM.

Como se observa en las Figs. 22 y 23, los resultados obtenidos por ambos métodos numéricos tienen una buena correspondencia.

VII.1.2. Intensidad del campo total de una PCW de conductor perfecto

En las Figs. 24 y 25 se muestran los resultados obtenidos de las intensidades de los campos cercano totales calculados con el IEM y FEM para las frecuencias $\omega_r = 1.3843$, $\omega_r = 1.5$, $\omega_r = 2.1371$ y $\omega_r = 3.0$. La intensidad del campo total corresponde a la suma de intensidades del campo incidente y esparcido.

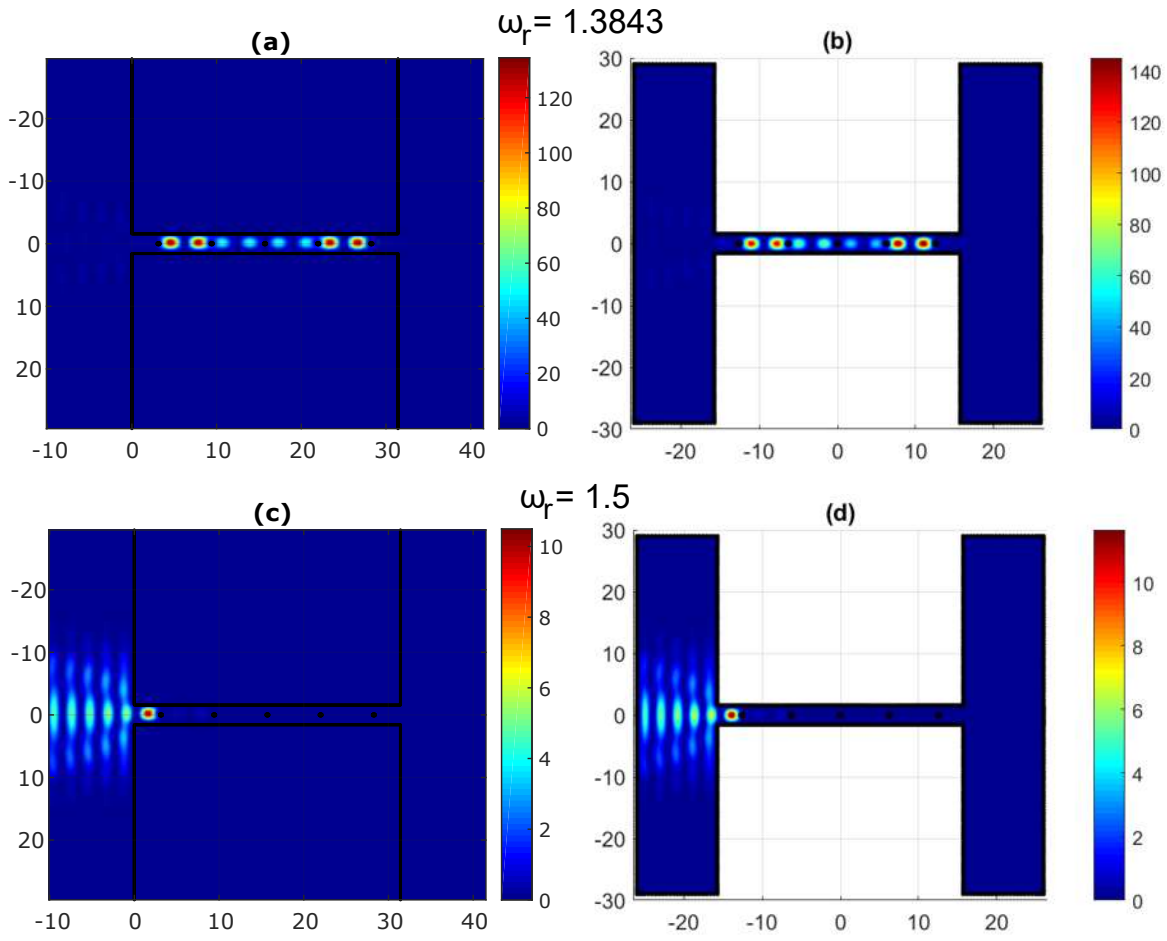

Figura 24: Intensidades del campo cercano total de una PCW de conductor perfecto mediante ((a) y (c)) IEM y ((b) y (d)) FEM.

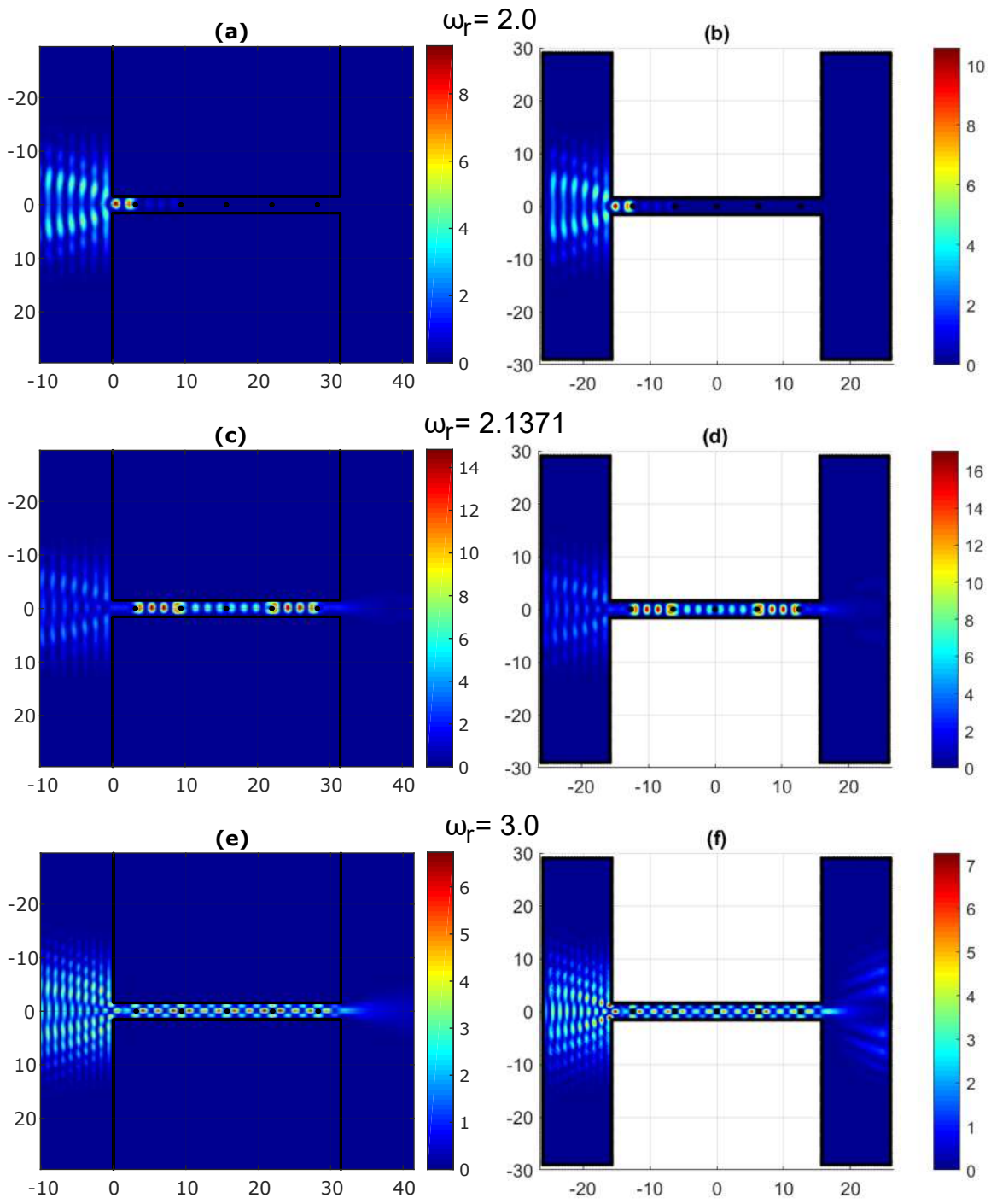

Figura 25: Intensidades del campo cercano total de una PCW de conductor perfecto mediante ((a), (c) y (e)) IEM y ((b), (d) y (f)) FEM.

De igual manera observamos en las Figs. 24 y 25, que los resultados obtenidos por ambos métodos tiene una buena correspondencia.

Capítulo VIII

CONCLUSIONES

En este capítulo mencionamos un breve resumen y en base a los resultados obtenidos enunciamos las conclusiones más importantes del trabajo.

En este trabajo nos centramos en estudiar distintos métodos numéricos así como de varias herramientas de programación para poder estudiar las guías de ondas en un tiempo razonable. Los métodos utilizados fueron el IEM y el FEM. Además, se utilizaron las herramientas para el cómputo en paralelo los protocolos y librerías: MPI, ScaLAPACK, OpenMP y CUDA.

El IEM parte del segundo teorema integral de Green permitiendo obtener un par de ecuaciones integrales acopladas que involucran como incógnitas el campo y su derivada normal evaluados en las fronteras o contornos involucrados. La discretización del sistema resulta en una ecuación matricial inhomogénea cuya solución determina las funciones fuentes, con las que se puede obtener la distribución del campo electromagnético. De esta manera calculamos el campo cercano y la respuesta óptica de una PCW bidimensional de longitud finita de conductor perfecto y real.

El FEM está basado en la reformulación del problema en forma variacional y solución del espacio de soluciones por otro de dimensión finita. Esto se logra al realizar una partición del dominio en elementos de tamaño finito y buscar soluciones

mediante familias de funciones dependientes de un número finito de parámetros. De esta manera se pasa de un problema inicial a un sistema de ecuaciones que se puede resolver numéricamente.

En la primera parte de este trabajo se analizaron diferentes formas de programación en paralelo aplicado al IEM en combinación con IBCM y MD, usando tanto los procesadores como la tarjeta gráfica y una combinación de ambos; así como en la forma secuencial para hacer un comparativo del tiempo de cómputo variando el número de periodos de la PCW de conductor real. Para la forma secuencial se usaron las librerías de LAPACK (externas) y MKL (internas) del compilador de Intel. Las otras formas de programación con el paradigma de paralelización fueron usando: las librerías de MPI con comunicación Colectiva (para este caso la parte que se paraleliza es el ciclo principal que se encarga de calcular la respuesta óptica de la PCW de conductor real para las distintas frecuencias), las librerías de OpenMP (para la construcción de la matriz), las librerías de ScaLAPACK (para la construcción e inversión de la matriz), las librerías internas de CUDA para la tarjeta gráfica (para la construcción e inversión de la matriz) y para la combinación de la GPU y CPU se utilizó CUDA para la construcción de las submatrices en la tarjeta gráfica y ScaLAPACK para la inversión de la matriz en los procesadores.

Las pruebas que se consideraron en este trabajo, fue tomando en cuenta la variación del número de periodos de 5 hasta 50 produciendo rangos de matrices de 3489×3489 hasta 15382×15382. En particular, para la matriz de rango 12740×12740, con la versión secuencial usando la librería MKL de Intel se obtuvo una rapidez de 74 veces más con respecto a la versión secuencial de LAPACK. En relación a las versiones paralelas con los procesadores de la CPU, usando OpenMP se tuvo una rapidez de 107 veces, para ScaLAPACK se tuvo una rapidez de 113 veces y para la versión de MPI se tuvo una rapidez de 154 veces más rápido. En cambio para la versión que utiliza la GPU se logró

una rapidez de 1406 veces más rápido. Por otro lado, para la versión que combina la GPU con la CPU se tuvo una rapidez de 164 veces más rápido que la versión secuencial utilizando LAPACK.

A pesar de que los resultados muestran que la forma más optima para programar en paralelo es hacerlo en la GPU; se tiene que la tarjeta GeForce GTX 1060 con 6 GB en VRAM presenta una desventaja, la cual es en el tamaño de la memoria que nada más permite invertir matrices complejas de rango 28000×28000 en precisión simple. En cambio usando los 4 procesadores y 16 GB en RAM con ScaLAPACK se puede invertir matrices complejas de rango 44000×44000. El protocolo de MPI (colectiva) también tiene un alto costo ya que utilizan la misma cantidad de memoria RAM en cada uno de los procesadores utilizados; es decir la cantidad de memoria total es el producto del número de procesadores por la memoria que utilizaría un programa secuencial bajo las mismas condiciones. En MPI sólo se puede invertir matrices complejas de rango 22000×22000, en cambio ScaLAPACK sólo utiliza la misma cantidad de memoria que un programa secuencial, en cambio al combinar la GPU y la CPU podemos invertir matrices complejas de rango 28000×28000 con una rapidez muy similar a la obtenida con MPI.

Por lo tanto, la conclusión principal de la primera parte de este trabajo es que con estas técnicas de programación en paralelo bajo el protocolo de MPI, ScaLAPACK y CUDA permiten modelar sistemas complejos en tiempo de cómputo relativamente muy cortos.

Para la segunda parte de este trabajo se utilizó el FEM en su forma débil de Galerkin en combinación con la técnica PML para resolver numéricamente la ecuación de Helmholtz para problemas en 1D y 2D con dominios abiertos. El primer problema con el que se trabajó fue en 1D y con condiciones de Dirichlet en una de las fronteras. Esto debido a la facilidad de encontrar la solución analítica y así tener un punto de

comparación respecto a la solución obtenida con el método numérico. La prueba realizada en este primer problema consistió en calcular el valor de función solución en la región de interés variando la cantidad de elementos que componían el espacio. Para hacer el comparativo entre la solución numérica y la solución analítica se eligió el punto donde se une el espacio físico con el material PML. Con 15 elementos se obtuvo un error del 287 %, para 30 elementos se obtuvo un error del 135 %, para 150 elementos se obtuvo un error del 0.15 % y finalmente para 1500 elementos se obtuvo un error del 0.0015 %. Como podemos observar, con una cantidad relativamente pequeña de elementos pudimos obtener una buena aproximación a la solución analítica del problema en 1D. El segundo problema consistió en calcular el campo producido por una fuente emitiendo en el espacio en 2D de manera numérica y comparándolo con la solución analítica la cual es conocida. Los resultados obtenidos por el método numérico tuvo una buena correspondencia con la solución analítica. El tercer problema fue muy similar al segundo problema, salvo que la región de interés esta dividida en dos regiones, donde cada una tiene un índice de refracción distinto. Los resultados obtenidos por el método numérico tuvo una buena correspondencia con lo predicho por la Ley de Snell. Ya que se puede apreciar que el frente de onda cambio su ángulo de propagación al cambiar de un medio a otro; así como se aprecia la interferencia producida por las ondas reflejadas por la transición de un medio a otro.

La conclusión principal para esta segunda parte de este trabajo es que el FEM es un buen método numérico general para la aproximación de ecuaciones diferenciales, el cual puede ser adaptado a problemas más particulares como fue nuestro caso: resolver la ecuación de ondas con dominios abiertos utilizando la técnica PML. Dicha técnica fue muy útil para poder adaptar nuestro problema de dominios abiertos a un dominio finito, sin tener que recurrir a técnicas que hacen uso de series para aproximar la solución en la frontera exterior como *Dirichlet-to-Neumann* entre otras.

Para la tercera parte de este trabajo se utilizó el IEM y FEM en su forma débil de Galerkin en combinación con la técnica PML para calcular el campo cercano de una PCW de conductor perfecto iluminada por un haz Gaussiano a distintas frecuencias. Los resultados obtenidos al calcular el campo reflejado y total por ambos métodos mostraron una buena correspondencia.

Por lo tanto, la conclusión general de este trabajo de investigación es que tanto el IEM y el FEM son excelentes métodos para modelar las PCWs, cada uno con sus ventajas y desventajas respecto al otro. Con IEM se pueden calcular el campo lejano de las PCWs en tiempos de cómputo pequeños. En cambio con el FEM se puede calcular el campo cercano de las PCWs de una manera muy rápida y precisa. Como futuro trabajo se pretende usar FEM para calcular el campo lejano de una PCW y combinarlo con las las técnicas de programación en paralelo vistas en este trabajo y así lograr reducir su tiempo de ejecución.

Apéndice A

ALGORITMOS AUXILIARES PARA CALCULAR LA RESPUESTA ÓPTICA DE UNA PCW DE CONDUCTOR REAL UTILIZANDO SCALAPACK

En este apartado se muestran los códigos de las rutinas que fueron utilizadas para calcular la respuesta óptica de una PCW de conductor real utilizando ScaLAPACK.

A.1. Algoritmo "DISTRIBUCIONCICLICA"

Esta rutina calcula los índices de las columnas o de las filas para realizar la distribución cíclica.

```
**********************************************************************
SUBROUTINE DISTRIBUCIONCICLICA ( M, NP, N, NB )

  IMPLICIT        NONE
  INTEGER         NP,N,NB
  INTEGER         M(NP,N)
  INTEGER         C,CO,CONTADOR,I

  C = 1
  CONTADOR = 0
  DO WHILE ( C.LE.N )
    CO = 1
```

```
    DO WHILE ( CO.LE.NP )
      I = 1
      DO WHILE ( I.LE.NB )
        IF ( C.LE.N ) THEN
          M (CO, I + ( CONTADOR*NB ) ) = C
          C = C + 1
        ELSE
          I = NB + 1
          CO = NP + 1
        END IF
        I = I + 1
      END DO
      CO = CO + 1
    END DO
    CONTADOR = CONTADOR + 1
  END DO
  RETURN
END

**********************************************************************
```

A.2. Algoritmo "MATINIT"

Esta rutina llena las submatrices *ME*, *EI* y las distribuye a cada proceso.

```
**********************************************************************

SUBROUTINE MATINIT( AA, DESCA, B, DESCB, N, NPC, NPR, TC, TR, NPT,&
  NANG, SF, SG, WNUM, G, D, DSF, DSG, DDSF, DDSG, DELTA, THETA0, LAMBD,&
  FDO, CIMP, NC, POL)

  IMPLICIT NONE

  INTEGER            DESCA( * ), DESCB( * )
  INTEGER            N, NPC, NPR, NPT, NANG, FDO, POL
  REAL               WNUM, G, D, LAMBD
  REAL               SF(NPT), SG(NPT), THETA0(NANG), DSF(NPT),&
                     DSG(NPT), DDSF(NPT), DELTA, DDSG(NPT)
  COMPLEX            AA( * ), B( * ), CIMP(NPT), TEM, NC
  INTEGER            CTXT_, LLD_, X, Y
  PARAMETER          ( CTXT_ = 2, LLD_ = 9 )
  INTEGER            ICTXT, MXLLDA, MXLLDB, MYCOL, MYROW, NPCOL, NPROW
  EXTERNAL           BLACS_GRIDINFO
```

```
  LOGICAL            BOL1, BOL2
  INTEGER            TC(NPC,N), TR(NPR,N)

  ICTXT = DESCA( CTXT_ )
  CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
  MXLLDA = DESCA( LLD_ )
  MXLLDB = DESCB( LLD_ )
  BOL1 = .TRUE.
  X = 1
  DO WHILE(BOL1)
    BOL2=.TRUE.
    Y=1
    DO WHILE(BOL2)
      CALL MATE( TEM, SF, DSF, DDSF, SG, DSG, DDSG, CIMP, NC, WNUM, &
      DELTA, NPT, POL, TR(MYROW+1,Y), TC(MYCOL+1,X))
      AA(Y+(X-1)*MXLLDA) = TEM

      IF(MYCOL.EQ.0 .AND.X.LE.NANG)THEN
        CALL EINUMNR(TEM,NPT,NANG,SF,SG,THETA0,LAMBD,G,D,FD0,WNUM, &
        TR(MYROW+1,Y),TC(MYCOL+1,X))
        B(Y+(X-1)*MXLLDA) = TEM
      END IF
      Y = Y+1
      IF(TR(MYROW+1,Y).EQ.0)THEN
        BOL2 = .FALSE.
      END IF
    END DO
    X = X+1
    IF(TC(MYCOL+1,X).EQ.0)THEN
      BOL1 = .FALSE.
    END IF
  END DO
  RETURN
END

**********************************************************************
```

A.3. Algoritmo "ORDENA"

Esta rutina es la encargada de ordenar la solución que produce cada procesador.

```
**********************************************************************

SUBROUTINE ORDENA (BTEM, B, MYROW, MYCOL, N, M, MXLLDB, MXRHSC, TR,
```

```
$           NPROW )

  IMPLICIT       NONE

  INTEGER        MYROW, MYCOL, N, M, MXLLDB, MXRHSC, NPROW, CON
  INTEGER        TR ( NPROW, N )
  COMPLEX        B ( MXLLDB, MXRHSC ), BTEM<( N, M )
  LOGICAL        BOL

  CON = 1
  BOL = .TRUE.

  DO WHILE ( BOL )
    IF (TR(MYROW+1,CON).NE.0) THEN
      BTEM ( TR ( MYROW + 1, CON ), M ) = B ( CON, MXRHSC )
      CON = CON + 1
    ELSE
      BOL = .FALSE.
    END IF
  END DO
  RETURN
END
```

****

Apéndice B

ALGORITMOS AUXILIARES PARA CALCULAR LA RESPUESTA ÓPTICA DE UNA PCW UTILIZANDO CUDA

En este apartado se muestran los códigos de las rutinas que fueron utilizadas para calcular la respuesta óptica de una PCW de conductor real utilizando CUDA.

B.1. Algoritmo "matmYeinumnr"

Esta rutina es la encargada de llenar las matrices ME y EI en forma paralela usando la GPU.

```
**********************************************************************

__global__ void mateYeinumnr_kernel( cuComplex* ME, float* JI,
  float* JI1, float* JI2, float* ETA, float* ETA1, float* ETA2,
  cuComplex* CIMP, float WNUM, float DELTJ, int NPT, int POL,
  cuComplex* EI, int NANG, float* THETA0, float G, float D ){

  float nume, arg1, dq, q, alpha0, Aqkp, arg, pi, kp, x1, x3;
  float wnum2, delts, rij, arg2, fj;
  cuComplex im1, sum, fact1, cotr, hh0, hh1, otr, t;
  int i, j, nq, tem;

  pi = 3.141592654f;
  im1.x = 0.0;
```

```
im1.y = 1.0;
nume = expf(1.0);
wnum2 = WNUM*WNUM;
otr.x = 0.0;
otr.y = 0.0;

i=threadIdx.x + blockIdx.x * blockDim.x;
j=threadIdx.y + blockIdx.y * blockDim.y;

if ( i < NPT ){
  if ( j < NPT ){
    delts = DELTJ;
    fact1.x = im1.x*delts/4.0;
    fact1.y = im1.y*delts/4.0;
    if( i != j ){
      rij = sqrtf((JI[i]-JI[j])*(JI[i]-JI[j])+
            (ETA[i]-ETA[j])*(ETA[i]-ETA[j]));
      arg1 = WNUM*rij;
      arg2 = ETA1[j]*(JI[i]-JI[j])-JI1[j]*(ETA[i]-ETA[j]);
      hh0.x = j0f(arg1);
      hh0.y = y0f(arg1);
      hh1.x = j1f(arg1)/arg1;
      hh1.y = y1f(arg1)/arg1;
      //pol s
      t = Cdiv(hh1,CIMP[j]);
      t.x = hh0.x+wnum2*arg2*t.x;
      t.y = hh0.y+wnum2*arg2*t.y;
      ME[i+j*NPT] = Cmul(fact1,t);
    }
    else{
      arg = WNUM*delts/(2.0*nume);
      t.x = j0f(arg);
      t.y = y0f(arg);
      cotr = Cmul(fact1,t);
      otr.x = 0.5-delts*(JI1[i]*ETA2[i]-JI2[i]*ETA1[i])/(4.0*pi);
      otr.y = 0.0;
      //pol s
      t = Cdiv(otr,CIMP[i]);
      ME[i+j*NPT].x = cotr.x+t.x;
      ME[i+j*NPT].y = cotr.y+t.y;
    }
  }
  else{
    if ( j-NPT < NANG ){
      nq = 721;
```

```
        dq = 2.0*WNUM/((nq*1.0)-1.0);
        kp = WNUM*sin(THETA0[(j-NPT)]);
        x1 = JI[i];
        x3 = ETA[i]-D;
        sum.x = 0.0;
        sum.y = 0.0;
        for (tem=1; tem<=nq-2; tem++) {
          fj = tem*1.0;
          q = -WNUM+fj*dq;
          alpha0 = sqrtf(WNUM*WNUM-q*q);
          Aqkp = sqrtf(pi)*G*expf(-(G*(q-kp)/2.0)*(G*(q-kp)/2.0));
          arg = q*x1-alpha0*x3;
          t.x = im1.x*arg;
          t.y = im1.y*arg;
          sum.x = sum.x+Aqkp*Cexp(t).x;
          sum.y = sum.y+Aqkp*Cexp(t).y;
        }
        EI[i+(j-NPT)*NPT].x = sum.x*dq/(2.0*pi);
        EI[i+(j-NPT)*NPT].y = sum.y*dq/(2.0*pi);
      }
    }
  }
}
```

****

Apéndice C

ALGORITMOS AUXILIARES PARA CALCULAR LA RESPUESTA ÓPTICA DE UNA PCW UTILIZANDO SCALAPACK CON CUDA

En este apartado se muestran los códigos de las rutinas que fueron utilizadas para calcular la respuesta óptica de una PCW de conductor real utilizando CUDA.

C.1. Algoritmo "DISTRIBUCIONCICLICA"

Esta rutina calcula los índices de las columnas o de las filas para realizar la distribución cíclica.

```
**************************************************************

int *DISTRIBUCIONCICLICA ( int *T, int NP, int N, int NB){
  int c,co,i;
  int contador;

  c = 1;
  contador = 0;
  do{
    co = 0;
    do{
      i = 0;
      do{
```

```
        if( c <= N ){
          T[co+NP*(i+contador*NB)] = c;
          c++;
        }
        else{
          i = NB+1;
          co = NP+1;
        }
        i++;
      }while( i < NB );
      co++;
    }while( co < NP );
    contador++;
  }while( c <= N );
  return 0;
}
```

****

C.2. Algoritmo "MAINIT_CUDA"

Esta rutina se utiliza como puente entre la CPU y la GPU.

****

```
extern "C" void MATINIT_CUDA(cuComplex* A_LOCAL, cuComplex* B_LOCAL,
  cuComplex* A_LOCAL_D, cuComplex* B_LOCAL_D, cuComplex* cimp_h,
  cuComplex* cimp_d, cuComplex* v, float* sf, float *sf_d, float* dsf,
  float* dsf_d, float* ddsf, float* ddsf_d, float* sg, float* sg_d,
  float* dsg, float* dsg_d, float* ddsg, float* ddsg_d, float* theta0,
  float* theta0_d, int* DESCA, int* DESCA_D, int* DESCB, int* DESCB_D,
  int* TR, int* TR_D, int* TC, int* TC_D, int npt, int nang, int fdo,
  int pol, int NPROC_ROWS, int NPROC_COLS, int NBRHS, int MXLOCR,
  int MXLOCC, int MXAM, int MXAN, int MXBM, int MYROW, int MYCOL){

  cudaMemcpy(sf_d,sf,npt*sizeof(float),cudaMemcpyHostToDevice);
  cudaMemcpy(sg_d,sg,npt*sizeof(float),cudaMemcpyHostToDevice);
  cudaMemcpy(dsf_d,dsf,npt*sizeof(float),cudaMemcpyHostToDevice);
  cudaMemcpy(dsg_d,dsg,npt*sizeof(float),cudaMemcpyHostToDevice);
  cudaMemcpy(ddsf_d,ddsf,npt*sizeof(float),cudaMemcpyHostToDevice);
  cudaMemcpy(ddsg_d,ddsg,npt*sizeof(float),cudaMemcpyHostToDevice);
  cudaMemcpy(theta0_d,theta0,nang*sizeof(float),cudaMemcpyHostToDevice);
  cudaMemcpy(cimp_d,cimp_h,npt*sizeof(cuComplex),
```

```
  cudaMemcpyHostToDevice);
  cudaMemcpy(DESCA_D,DESCA,9*sizeof(int),cudaMemcpyHostToDevice);
  cudaMemcpy(DESCB_D,DESCB,9*sizeof(int),cudaMemcpyHostToDevice);
  cudaMemcpy(TR_D,TR,NPROC_ROWS*MXLOCR*sizeof(int),
  cudaMemcpyHostToDevice);
  cudaMemcpy(TC_D,TC,NPROC_COLS*MXLOCC*sizeof(int),
  cudaMemcpyHostToDevice);

  dim3 blockSize (32,32,1);
  dim3 gridSize (floor(MXAM/32)+32,floor(MXAN/32)+32,1);

  Construye<<<gridSize,blockSize>>>
    (A_LOCAL_D,B_LOCAL_D,cimp_d,sf_d,dsf_d,ddsf_d,sg_d,dsg_d,ddsg_d,
    DESCA_D,DESCB_D,TR_D,TC_D,theta0_d,npt,nang,fdo,pol,v[0].x,
    v[1].x,v[2].x,v[3].x,v[4],NPROC_ROWS,NPROC_COLS,MYROW,
    MYCOL,MXLOCR,MXLOCC,NBRHS);

  cudaDeviceSynchronize();
  cudaMemcpy(B_LOCAL,B_LOCAL_D,MXBM*NBRHS*sizeof(cuComplex),
  cudaMemcpyDeviceToHost);

  cudaMemcpy(A_LOCAL,A_LOCAL_D,MXAM*MXAN*sizeof(cuComplex),
  cudaMemcpyDeviceToHost);
}
```

****

C.3. Algoritmo "Construye"

Esta rutina es la encargada de llenar las submatrices A_LOCAL y B_LOCAL utilizando la GPU.

****

```
__global__ void Construye( cuComplex* A, cuComplex* B, cuComplex* cimp,
  float* sf ,float* dsf, float* ddsf, float* sg, float* dsg,
  float* ddsg, int* DESCA, int* DESCB, int* TR, int* TC, float* theta0,
  int npt, int nang, int fdo, int pol, float delta, float wnum, float g,
  float d, cuComplex nc, int NPR, int NPC, int MYROW, int MYCOL,
  int MXLOCR, int MXLOCC, int NBRHS){

  int R,C,i,j,M,MXAL;
  cuComplex tem;
```

```
  tem.x = 0.0;
  tem.y = 0.0;
  M = DESCA[2];
  MXAL = DESCA[8];

  R = threadIdx.x + blockIdx.x * blockDim.x;
  C = threadIdx.y + blockIdx.y * blockDim.y;
  if( TC[MYCOL+C*NPC] != 0 && (MYCOL+C*NPC) < (NPC*MXLOCC)){
    if( TR[MYROW+R*NPR] != 0&& (MYROW+R*NPR) < (NPR*MXLOCR)){
      i = TR[MYROW+R*NPR]-1;
      j= TC[MYCOL+C*NPC]-1;
      tem=mate(sf,dsf,ddsf,sg,dsg,ddsg,cimp,nc,wnum,delta,M,pol,i,j);
      A[R+C*MXAL] = tem;
      if(TC[MYCOL+C*NPC] <= NBRHS){
        tem = einumnr(M,NBRHS,sf,sg,theta0,g,d,fdo,wnum,i,j);
        B[R+C*MXAL] = tem;
      }
    }
  }
}
```

****

C.4. Algoritmo "mate"

Esta rutina es la encargada de calcular el valor para submatriz A_LOCAL en la posición i, j.

****

```
__device__ cuComplex mate(float* JI, float* JI1, float* JI2, float* ETA,
  float* ETA1, float* ETA2, cuComplex *CIMP, cuComplex nc, float WNUM,
  float DELTJ, int NPT, int pol, int i, int j){

  float nume,arg1,arg,pi;
  float wnum2,delts,rij,arg2;
  cuComplex im1,fact1,cotr,hh0,hh1,otr,t;

  pi = 3.141592654f;
  im1.x = 0.0;
  im1.y = 1.0;
  nume = expf(1.0);
```

```
  wnum2 = WNUM*WNUM;
  otr.x = 0.0;
  otr.y = 0.0;

  if( i < NPT ){
    if( j < NPT ){
      delts = DELTJ;
      fact1.x = im1.x*delts/4.0;
      fact1.y = im1.y*delts/4.0;
      if( i != j ){
        rij = sqrt((JI[i]-JI[j])*(JI[i]-JI[j])+
          (ETA[i]-ETA[j])*(ETA[i]-ETA[j]));
        arg1 = WNUM*rij;
        arg2 = ETA1[j]*(JI[i]-JI[j])-JI1[j]*(ETA[i]-ETA[j]);
        hh0.x = j0f(arg1);
        hh0.y = y0f(arg1);
        hh1.x = j1f(arg1)/arg1;
        hh1.y = y1f(arg1)/arg1;
        //pol s
        t = Cdiv(hh1,CIMP[j]);
        t.x = hh0.x+wnum2*arg2*t.x;
        t.y = hh0.y+wnum2*arg2*t.y;
        t = Cmul(fact1,t);
      }
      else{
        arg = WNUM*delts/(2.0*nume);
        t.x = j0f(arg);
        t.y = y0f(arg);
        cotr = Cmul(fact1,t);
        otr.x = 0.5-delts*(JI1[i]*ETA2[i]-JI2[i]*ETA1[i])/(4.0*pi);
        otr.y=0.0;
        //pol s
        t = Cdiv(otr,CIMP[i]);
        t.x = cotr.x+t.x;
        t.y = cotr.y+t.y;
      }
    }
  }
  return t;
}

**********************************************************************
```

C.5. Algoritmo "einumnr"

Esta rutina es la encargada de calcular el valor para submatriz B_LOCAL en la posición i, j.

```
**********************************************************************

__device__ cuComplex einumnr(int NPT, int NANG, float *JI, float *ETA,
  float *THETA0, float G, float D, int fdo, float WNUM, int i, int j){

  float dq,q,alpha0,Aqkp,arg,pi,kp,x1,x3,fj;
  cuComplex im1,sum,t;
  int nq,tem;

  pi = 3.141592654f;
  im1.x = 0.0;
  im1.y = 1.0;

  if( j < NANG ){
    nq = 721;
    dq = 2.0*WNUM/((nq*1.0)-1.0);
    kp = WNUM*sin(THETA0[j]);
    if( i < NPT ){
      x1 = JI[i];
      x3 = ETA[i]-D;
      sum.x = 0.0;
      sum.y = 0.0;
      for ( tem=1; tem<=nq-2; tem++) {
        fj = tem*1.0;
        q = -WNUM+fj*dq;
        alpha0 = sqrt(WNUM*WNUM-q*q);
        Aqkp = sqrt(pi)*G*exp(-(G*(q-kp)/2.0)*(G*(q-kp)/2.0));
        arg = q*x1-alpha0*x3;
        t.x = im1.x*arg;
        t.y = im1.y*arg;
        sum.x = sum.x+Aqkp*Cexp(t).x;
        sum.y = sum.y+Aqkp*Cexp(t).y;
      }
      t.x = sum.x*dq/(2.0*pi);
      t.y = sum.y*dq/(2.0*pi);
    }
  }
  return t;
}
```

****

C.6. Algoritmo "ORDENA"

Esta rutina es la encargada de ordenar la solución que produce cada procesador.

****

```
int *ORDENA( float complex *B_GLOBAL, float complex *B_LOCAL, int MYROW,
  int MYCOL, int M, int N, int MXLLDB, int MXRHSC, int *TR, int *TC,
  int NPROW, int NPCOL){

  int R,C;
  bool BOL;
  BOL = true;
  R = 0;
  do{
    if( TR[MYROW+R*NPROW] != 0){
      for( C=0; C<MXRHSC; C++){
        B_GLOBAL[(TR[MYROW+R*NPROW]-1)+(TC[MYCOL+C*NPCOL]-1)*M] =
        B_LOCAL[R+C*MXLLDB];
      }
      R = R+1;
    }
    else{
      BOL = false;
    }
  }while ( BOL );
  return 0;
}
```

****

Apéndice D

ALGORITMOS AUXILIARES PARA EL MÉTODO DE ELEMENTOS FINITOS EN COMBINACIÓN CON LA TÉCNICA PML PARA LA ECUACIÓN DE HELMHOLTZ 1D

En este apartado se muestran los códigos de los algoritmos auxiliares que fueron utilizados para el *Programa 5*.

D.1. Algoritmo "IntegralK1"

Este algoritmo calcula las integrales para la matriz auxiliar Ke para el elemento e en la región $DPML$.

****

```
function f=IntegralK1(i,j,x,disx,dpmlx,k)
  h=x(2)-x(1);
  DN1=@(x)(-1/h);
  DN2=@(x)(1/h);
  img=complex(0,1);
  c=dpmlx;
  %sigma=@(x)c./(dpmlx-x+disx);
  %gamma=@(x)1+(img/k).*sigma(x);
  %usgamma=1./gamma;
```

```
  usgamma=@(x)(k.*(dpmlx+disx-x)./(k.*(dpmlx+disx-x)+img*c));
  if(i==1)
    if(j==1)
      fun=@(x)usgamma(abs(x)).*DN1(x).*DN1(x);
      f=integral(fun,x(1),x(2));
    end
    if(j==2)
      fun=@(x)usgamma(abs(x)).*DN1(x).*DN2(x);
      f=integral(fun,x(1),x(2));
    end
  end
  if(i==2)
    if(j==1)
      fun=@(x)usgamma(abs(x)).*DN2(x).*DN1(x);
      f=integral(fun,x(1),x(2));
    end
    if(j==2)
      fun=@(x)usgamma(abs(x)).*DN2(x).*DN2(x);
      f=integral(fun,x(1),x(2));
    end
  end
end
```

****

D.2. Algoritmo "IntegralM1"

Este algoritmo calcula las integrales para la matriz auxiliar Me para el elemento e en la región $DPML$.

****

```
function f=IntegralM1(i,j,x,disx,dpmlx,k)
  h=x(2)-x(1);
  N1=@(x)(-1/h)*(x-x(1))+1;
  N2=@(x)(1/h)*(x-x(2))+1;
  img=complex(0,1);
  c=dpmlx;
  %sigma=@(x)c./(dpmlx-x+disx);
  %gamma=@(x)1+(img/k).*sigma(x);
  %usagamma=gamma(sigma(x));
```

```
  usgamma=@(x)(k.*(dpmlx-x+disx)+img*c)./(k.*(dpmlx-x+disx));
  if(i==1)
    if(j==1)
      fun=@(x)usgamma(abs(x)).*N1(x).*N1(x);
      f=integral(fun,x(1),x(2));
    end
    if(j==2)
      fun=@(x)usgamma(abs(x)).*N1(x).*N2(x);
      f=integral(fun,x(1),x(2));
    end
  end
  if(i==2)
    if(j==1)
      fun=@(x)usgamma(abs(x)).*N2(x).*N1(x);
      f=integral(fun,x(1),x(2));
    end
    if(j==2)
      fun=@(x)usgamma(abs(x)).*N2(x).*N2(x);
      f=integral(fun,x(1),x(2));
    end
  end
end
```

****

Apéndice E

ALGORITMOS AUXILIARES PARA MÉTODO DE ELEMENTOS FINITOS EN COMBINACIÓN CON LA TÉCNICA PML PARA LA ECUACIÓN DE HELMHOLTZ 2D

En este apartado se muestran los códigos de los algoritmos auxiliares que fueron utilizados para el *Programa 6,7 y 8*.

E.1. Algoritmo "Correccion_orientacion"

Este algoritmo corrige la orientación de los triángulos.

```
**************************************************************

function [t,x,y,J,nodes]=Correccion_orientacion(t,e,p)
  nodes=t(e,:);
  x=[p(nodes(1),1),p(nodes(2),1),p(nodes(3),1)];
  y=[p(nodes(1),2),p(nodes(2),2),p(nodes(3),2)];
  J=(x(2)-x(1))*(y(3)-y(1))-(x(3)-x(1))*(y(2)-y(1));

  if(J<0)
    temporal=t(e,2);
    t(e,2)=t(e,3);
    t(e,3)=temporal;
    nodes=t(e,:);
```

```
    x=[p(nodes(1),1),p(nodes(2),1),p(nodes(3),1)];
    y=[p(nodes(1),2),p(nodes(2),2),p(nodes(3),2)];
    J=(x(2)-x(1))*(y(3)-y(1))-(x(3)-x(1))*(y(2)-y(1));
  end
end
```

****

E.2. Algoritmo "Revisa_Ubicacion"

Este algoritmo revisa la ubicación de los triángulos; regresa 1 si el triángulo está en la región de estudio y regresa 2 si el triángulo está en la región $DPML$.

****

```
function val=Revisa_Ubicacion(x,y,disx,disy)
  val=2;
  if(x(1)<=disx && x(1)>=-disx && x(2)<=disx && x(2)>=-disx &&...
     x(3)<=disx && x(3)>=-disx )
    if(y(1)<=disy && y(1)>=-disy && y(2)<=disy && y(2)>=-disy &&...
     y(3)<=disy && y(3)>=-disy )
      val=1;
    end
  end
end
```

****

E.3. Algoritmo "Region_Estudio"

Este algoritmo calcula las matrices Ke y Me para la región de estudio.

****

```
function [Ke,Me]=Region_Estudio(x,y,J,k0)
  k11=(x(3)-x(2))^2+(y(2)-y(3))^2;
  k12=(x(1)-x(3))*(x(3)-x(2))+(y(2)-y(3))*(y(3)-y(1));
  k21=(x(1)-x(3))*(x(3)-x(2))+(y(2)-y(3))*(y(3)-y(1));
  k13=(x(2)-x(1))*(x(3)-x(2))+(y(1)-y(2))*(y(2)-y(3));
  k31=(x(2)-x(1))*(x(3)-x(2))+(y(1)-y(2))*(y(2)-y(3));
  k22=(x(1)-x(3))^2+(y(3)-y(1))^2;
```

```
  k23=(x(2)-x(1))*(x(1)-x(3))+(y(1)-y(2))*(y(3)-y(1));
  k32=(x(2)-x(1))*(x(1)-x(3))+(y(1)-y(2))*(y(3)-y(1));
  k33=(x(2)-x(1))^2+(y(1)-y(2))^2;

  Ke= -(1/(2*J))*[k11 k12 k13;k21 k22 k23; k31 k32 k33];
  Me=k0^2*(J/24)*[2 1 1;1 2 1;1 1 2];
end
```

****

E.4. Algoritmo "Region_PML"

Este algoritmo calcula las matrices Ke y Me para la región PML.

****

```
function [Ke,Me]=Region_PML(x,y,J,k,dpmlx,dpmly,disx,disy)
  k11=Integral_K(1,1,x,y,dpmlx,dpmly,disx,disy,k);
  k12=Integral_K(1,2,x,y,dpmlx,dpmly,disx,disy,k);
  k13=Integral_K(1,3,x,y,dpmlx,dpmly,disx,disy,k);
  k21=k12;
  k22=Integral_K(2,2,x,y,dpmlx,dpmly,disx,disy,k);
  k23=Integral_K(2,3,x,y,dpmlx,dpmly,disx,disy,k);
  k31=k13;
  k32=k23;
  k33=Integral_K(3,3,x,y,dpmlx,dpmly,disx,disy,k);

  m11=Integral_M(1,1,x,y,dpmlx,dpmly,disx,disy,k);
  m12=Integral_M(1,2,x,y,dpmlx,dpmly,disx,disy,k);
  m13=Integral_M(1,3,x,y,dpmlx,dpmly,disx,disy,k);
  m21=m12;
  m22=Integral_M(2,2,x,y,dpmlx,dpmly,disx,disy,k);
  m23=Integral_M(2,3,x,y,dpmlx,dpmly,disx,disy,k);
  m31=m13;
  m32=m23;
  m33=Integral_M(3,3,x,y,dpmlx,dpmly,disx,disy,k);


  Ke= -[k11 k12 k13;k21 k22 k23; k31 k32 k33];
  Me=k^2*(J)*[m11 m12 m13;m21 m22 m23; m31 m32 m33];
end
```

****

E.5. Algoritmo "Integral_K"

Este algoritmo calcula las integrales para la matriz auxiliar Ke para el elemento e en la región $DPML$.

****

```
function f=Integral_K(i,j,x,y,dpmlx,dpmly,disx,disy,k)
  x1=x(1);x2=x(2);x3=x(3);
  y1=y(1);y2=y(2);y3=y(3);
  a1=x2*y3-x3*y2; b1=y2-y3;   c1=x3-x2;
  a2=x3*y1-x1*y3; b2=y3-y1;   c2=x1-x3;
  a3=x1*y2-x2*y1; b3=y1-y2;   c3=x2-x1;
  tem=[x',y'];
  Pe=[ones(3,1),tem];
  A=abs(det(Pe))/2;
  pm=mean(tem);

  DN1X=@(x,y)(1/(2*A))*(b1);        DN1Y=@(x,y)(1/(2*A))*(c1);
  DN2X=@(x,y)(1/(2*A))*(b2);        DN2Y=@(x,y)(1/(2*A))*(c2);
  DN3X=@(x,y)(1/(2*A))*(b3);        DN3Y=@(x,y)(1/(2*A))*(c3);

  img=complex(0,1);
  sigmax=@(x)dpmlx./(dpmlx-x+disx);
  gammax=@(x)(x>=disx).*(1+(img/k).*sigmax(x))+(x<disx).*(1);

  sigmay=@(y)dpmly./(dpmly-y+disy);
  gammay=@(y)(y>=disy).*(1+(img/k).*sigmay(y))+(y<disy).*(1);

  if(i==1)
    if(j==1)
      fx=A*DN1X(pm(1)).*DN1X(pm(2)).*(gammay(abs(pm(2))./...
         gammax(abs(pm(1))));
      fy=A*DN1Y(pm(1)).*DN1Y(pm(2)).*(gammax(abs(pm(1))./...
         gammay(abs(pm(2))));
    end
    if(j==2)
      fx=A*DN1X(pm(1)).*DN2X(pm(2)).*(gammay(abs(pm(2))./...
         gammax(abs(pm(1))));
      fy=A*DN1Y(pm(1)).*DN2Y(pm(2)).*(gammax(abs(pm(1))./...
         gammay(abs(pm(2))));
    end
    if(j==3)
      fx=A*DN1X(pm(1)).*DN3X(pm(2)).*(gammay(abs(pm(2))./...
```

```
        gammax(abs(pm(1))));
    fy=A*DN1Y(pm(1)).*DN3Y(pm(2)).*(gammax(abs(pm(1))./...
        gammay(abs(pm(2))));
  end
end
if(i==2)
  if(j==1)
    fx=A*DN2X(pm(1)).*DN1X(pm(2)).*(gammay(abs(pm(2))./...
        gammax(abs(pm(1))));
    fy=A*DN2Y(pm(1)).*DN1Y(pm(2)).*(gammax(abs(pm(1))./...
        gammay(abs(pm(2))));
  end
  if(j==2)
    fx=A*DN2X(pm(1)).*DN2X(pm(2)).*(gammay(abs(pm(2))./...
        gammax(abs(pm(1))));
    fy=A*DN2Y(pm(1)).*DN2Y(pm(2)).*(gammax(abs(pm(1))./...
        gammay(abs(pm(2))));
  end
  if(j==3)
    fx=A*DN2X(pm(1)).*DN3X(pm(2)).*(gammay(abs(pm(2))./...
        gammax(abs(pm(1))));
    fy=A*DN2Y(pm(1)).*DN3Y(pm(2)).*(gammax(abs(pm(1))./...
        gammay(abs(pm(2))));
  end
end
if(i==3)
  if(j==1)
    fx=A*DN3X(pm(1)).*DN1X(pm(2)).*(gammay(abs(pm(2))./...
        gammax(abs(pm(1))));
    fy=A*DN3Y(pm(1)).*DN1Y(pm(2)).*(gammax(abs(pm(1))./...
        gammay(abs(pm(2))));
  end
  if(j==2)
    fx=A*DN3X(pm(1)).*DN2X(pm(2)).*(gammay(abs(pm(2))./...
        gammax(abs(pm(1))));
    fy=A*DN3Y(pm(1)).*DN2Y(pm(2)).*(gammax(abs(pm(1))./...
        gammay(abs(pm(2))));
  end
  if(j==3)
    fx=A*DN3X(pm(1)).*DN3X(pm(2)).*(gammay(abs(pm(2))./...
        gammax(abs(pm(1))));
    fy=A*DN3Y(pm(1)).*DN3Y(pm(2)).*(gammax(abs(pm(1))./...
        gammay(abs(pm(2))));
  end
end
```

```
  f=fx+fy;
end
```

****

E.6. Algoritmo "Integral_M"

Este algoritmo calcula las integrales para la matriz auxiliar Me para el elemento e en la región $DPML$.

****

```
function f=Integral_M(i,j,x,y,dpmlx,dpmly,disx,disy,k)
  x1=x(1);x2=x(2);x3=x(3);
  y1=y(1);y2=y(2);y3=y(3);

  img=complex(0,1);
  sigmax=@(x)dpmlx./(dpmlx-x+disx);
  gammax=@(x)(x>=disx).*(1+(img/k).*sigmax(x))+(x<disx).*(1);

  sigmay=@(y)dpmly./(dpmly-y+disy);
  gammay=@(y)(y>=disy).*(1+(img/k).*sigmay(y))+(y<disy).*(1);

  H1=@(s,t)1-s-t; H2=@(s,t)s; H3=@(s,t)t;

  if(i==1)
    if(j==1)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H1(s,t).*H1(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
    if(j==2)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H1(s,t).*H2(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
    if(j==3)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H1(s,t).*H3(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
  end
  if(i==2)
```

```
    if(j==1)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H2(s,t).*H1(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
    if(j==2)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H2(s,t).*H2(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
    if(j==3)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H2(s,t).*H3(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
  end
  if(i==3)
    if(j==1)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H3(s,t).*H1(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
    if(j==2)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H3(s,t).*H2(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
    if(j==3)
      fun1=@(s,t)(gammay(abs(y1+s.*(y2-y1)+t.*(y3-y1))).*...
           gammax(abs(x1+s.*(x2-x1)+t.*(x3-x1)))).*H3(s,t).*H3(s,t);
      f=(1/2)*(fun1(1/3,1/3));
    end
  end
end
```

****

E.7. Algoritmo "haz_gaussiano"

Este algoritmo calcula el valor del haz Gaussiano en la posición (x,y) y regresa su valor.

****

```
function f=haz_gaussiano(x,y,wnum,g,theta,d)
```

```
  im1=complex(0.0,1.0);
  zero=complex(0.0,0.0);
  gc=g*g;
  nq=721;
  dq=2.0*wnum/(double(nq)-1.0);
  kp=wnum*sin(theta);
  km=wnum*sin(-theta);
  sum=zero;
  x1=y;
  x3=-(x+d);
  for jj=1:nq-2
    fj=double(jj);
    q=-wnum+fj*dq;
    alpha0=sqrt(wnum*wnum-q*q);
    Aqkp=sqrt(pi)*g*exp(-(g*(q-kp)/2.0)^2);
    arg=q*x1-alpha0*x3;
    sum=sum+Aqkp*exp(im1*arg);
  end
  f=sum*dq/(2.0*pi);
end

*****************************************************************
```

Referencias

Adams, R. and Fournier, J. (2003). *Sobolev Spaces*, Vol. 140. Elsevier. Second edition.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. En *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), páginas 483–485, New York, NY, USA. ACM.

Berenger, J. . (1996). Perfectly matched layer for the fdtd solution of wave-structure interaction problems. *IEEE Transactions on Antennas and Propagation*, **44**(1): 110–117.

Berenger, J.-P. (1994). A perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phys.*, **114**(2): 185–200.

Bermudez, A., Hervella-Nieto, L., Prieto, A., and Rodrguez, R. (2006). An optimal finite-element/pml method for the simulation of acoustic wave propagation phenomena. *Variational Formulations in Mechanics: Theory and Applications*.

Bermúdez, A., Hervella-Nieto, L., Prieto, A., and Rodríguez, R. (2007). An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems. *Journal of Computational Physics*, **223**(2): 469 – 488.

Blackford, L. S., Choi, J., Cleary, A., D'Azeuedo, E., Demmel, J., Dhillon, I., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., and Whaley, R. C. (1997). *ScaLAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. ISBN 0-89871-397-8.

Brezis, H. (2011). *Funtional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, New York.

Cadien, K. C., Reshotko, M. R., Block, B. A., Bowen, A. M., Kencke, D. L., and Davids, P. (2005). Challenges for on-chip optical interconnects.

Centeno Jiménez, T. P. (2014). *Estructura de Bandas de Cristales Fotónicos en 2D con Superficies Rugosas Usando un Método Integral*. Tesis de Licenciatura, Facultad de Ciencias Físico Matemáticas de la UMSNH.

Chapman, B., Jost, G., Van der Pas, R., and Kuck, D. J. (2008). *Using OpenMP : portable shared memory parallel programming*. MIT Press, Cambridge, Mass., London.

Cook, S. (2013). *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. First edition.

Gedney, S. (1997). An anisotropic perfectly matched layer absorbing media for the truncation of fdtd lattice. *Antennas and Propagation, IEEE Transactions on*, **44**: 1630 – 1639.

Inan, U. and Marshall, R. (2011). *Numerical Electromagnetics: The FDTD Method*. Cambridge University Press.

Jackson, J. (1999). *Classical electrodynamics*. Alhambra.

Joannopoulos, J., Meade, R., Winn, J., Johnson, S., and Press, P. U. (2008). *Photonic Crystals: Molding the Flow of Light*. Princeton University Press. Second edition.

John, S. (1987). Strong localization of photons in certain disordered dielectric superlattices. *Physical review letters*, **58**: 2486–2489.

Kaeli, D. R., Mistry, P., Schaa, D., and Zhang, D. P. (2015). *Heterogeneous Computing with OpenCL 2.0*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. First edition.

Krauss, T., De La Rue, R., and Brand, S. (1996). Two-dimensional photonic-bandgap structures operating at near-infrared wavelengths. *Nature*, **383**.

Kwon, Y. W. and Bang, H. (1996). *Finite Element Method Using MATLAB*. CRC Press, Inc., Boca Raton, FL, USA. ISBN 0849396530. First edition.

Lax, P. D. and Milgram, A. N. (1954). Parabolic equations. en contributios to the theory of partial diferential equations. *Annals of Mathematics Studies*, (no. 33): 167–190. Princeton University Press, Princeton.

Lozano Trejo, E. (2017). *Estudio Numérico de la Respuesta Óptica en Guías de Ondas de Cristal Fotónico Usando Programación en Paralelo con los Protocolos de MPI y CUDA*. Tesis de Licenciatura, Facultad de Ciencias Físico Matemáticas de la UMSNH.

McKelvey, J. (1966). *Solid State and Semiconductor Physics*. Harper's physics series. Harper & Row. First edition.

Mendoza-Suárez, A. (1996). *Métodos rigurosos para esparcimiento de luz por superficies y medios estratificados con perfiles arbitrarios*. Tesis de Doctorado, CICESE.

Mendoza-Suárez, A. and Pérez-Aguilar, H. (2016). Numerical integral methods to study plasmonic modes in a photonic crystal waveguide with circular inclusions that involve a metamaterial. *Photonics*, **21**: 1–12.

Mendoza-Suárez, A., Villa-Villa, F., and Gaspar-Armenta, J. A. (2006). Numerical method based on the solution of integral equations for the calculation of the band structure and reflectance of one- and two-dimensional photonic crystals. *J. Opt. Soc. Am. B*, **23**(10): 2249–2256.

Mu, L., Wang, J., Ye, X., and Zhang, S. (2015). A weak galerkin finite element method for the maxwell equations. *Journal of Scientific Computing*, **65**(1): 363–386.

Pacheco, P. (2011). *An Introduction to Parallel Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pérez, H. I., Valencia, C. I., Méndez, E. R., and Sánchez-Gil, J. A. (2009). On the transmission of diffuse light through thick slits. *J. Opt. Soc. Am. B*, **26**(4): 909–918.

Pérez Hernández, E. (2015). *Estudio Numérico de la Propagación de la Luz en Guía de Ondas Periódicas y Onduladas Usando Programación en Paralelo.* Tesis de Licenciatura, Facultad de Ciencias Físico Matemáticas de la UMSNH.

Pérez, M. L. (2017). *Teorema de Lax-Milgram: origen, generalizaciones y aplicaciones.* Trabajo de Fin de grado, Departamente de Análisis Matemático de la Universidad de Granada.

Sánchez López, S. (2016). *Estudio Numérico de la Programación en Paralelo Usando el Protocolo Cuda Fortran.* Tesis de Licenciatura, Facultad de Ciencias Físico Matemáticas de la UMSNH.

Sibilia, C., Benson, T., Marciniak, M., and Szoplik, T. (2009). *Photonic Crystals: Physics and Technology*. Springer Milan. ISBN 9788847008441.

Sözüer, H. S., Haus, J. W., and Inguva, R. (1992). Photonic bands: Convergence problems with the plane-wave method. *Phys. Rev. B*, **45**: 13962–13972.

Teixeira, F. L. and Chew, W. C. (1998). A general approach to extend berenger's absorbing boundary condition to anisotropic and dispersive media. *IEEE Transactions on Antennas and Propagation*, **46**(9): 1386–1387.

Yablonovitch, E. (1987). Inhibited spontaneous emission in solid-state physics and electronics. *Physical review letters*, **58**: 2059–2062.

Álvarez Melis, D. (2011). *Teorema de Lax-Milgram: generalizaciones y aplicaciones.* Bsc Thesis. Instituto Tecnológico Autónomo de México.