UNIVERSIDAD DE SONORA

DEPARTAMENTO DE FÍSICA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES

TESIS DE DOCTORADO:

SIMULACIÓN COMPUTACIONAL DEL CAMPO ELECTROMAGNÉTICO EN NANOESTRUCTURAS UTILIZANDO EL MÉTODO DE DIFERENCIAS FINITAS EN EL DOMINIO DEL TIEMPO

Tesis enviada por Yohan Jasdid Rodríguez Viveros para la obtención del grado de Doctorado en Nanotecnología en la Universidad de Sonora

---

Director de Tesis:
Dr. Jesús Manzanares Martínez

Hermosillo, Sonora - 2015

Universidad de Sonora

Repositorio Institucional UNISON

Con todo mi cariño a mi hijos. A mis padres y
mis hermanos. A mi esposa y su familia.

Yohan J. Rodríguez Viveros

Resumen

Esta tesis presenta un análisis sobre el Método de Diferencias Finitas en el Dominio del Tiempo (FDTD) por sus siglas en inglés y su aplicación en nanoestructuras. En este trabajo se expone la realización de experimentos numéricos mediante tecnología computacional. Se construyen simulaciones de diversas situaciones físicas que permiten describir el comportamiento del campo electromagnético en estructuras en la escala del nanómetro. Este trabajo está consagrado a la transformación de las ecuaciones de Maxwell a ecuaciones en diferencias finitas que se pueden resolver numéricamente para crear un algoritmo computacional.

El primer capítulo presenta una introducción a los temas a tratar así como varios conceptos sobre nanotecnología. Se establecen los antecedentes y la historia, así como también se plantea el contexto general de la tesis y la relación que existe entre las múltiples disciplinas que abarca.

El segundo capítulo hace un recorrido de la teoría involucrada en la tesis. Se contempla el tratamiento analítico y matemático que incluye el desarrollo y resolución de las ecuaciones de Maxwell, hasta llegar a la ecuación de onda y la relación de dispersión. Se plantean sistemas y el procedimiento requerido para resolverlos.

El tercer capítulo describe el método FDTD. Se analizan las ecuaciones de Maxwell, se describe la forma de discretización a utilizar y como crear un algoritmo computacional capaz de resolver las ecuaciones. Se define el tratamiento analítico de las ecuaciones diferenciales, se plantean las fuentes y detectores y se define como aplicar la transformada de Fourier numéricamente.

El cuarto capítulo describe a Phoxonics, un software de simulación que se ha creado durante la investigación doctoral. Se ilustra con diagramas, conceptos y técnicas, como ha sido diseñada la arquitectura del software. Se toman en cuenta características de un diseño flexible, novedoso y potente que pueda brindar la capacidad de plantear y resolver múltiples sistemas de una manera sencilla y abstraer la complejidad en entidades computacionales.

El quinto capítulo expone la conversión de modos causado por doblamiento de guías de onda fotónicas donde se establece un sistema y se presenta un análisis teórico y computacional del fenómeno de conversión de modos. Se presentan los resultados cuantitativos descubiertos y la relación de estos con el doblamiento.

El sexto capítulo presenta las conclusiones y resultados de la investigación así como el trabajo a futuro.

Prefacio

"Sorprenderse, extrañarse, es comenzar a entender."
José Ortega y Gasset

El campo de la nanotecnología es un área creciente de la ciencia que se enfoca en el estudio de los fenómenos físicos en la escala del nanómetro. Esta tesis se relaciona directamente con la nanotecnología al enfocarse en el estudio de nanoestructuras utilizando como medio la simulación computacional.

La simulación computacional en la forma en que se expone resuelve las ecuaciones de Maxwell mediante un método de análisis llamado *Método de Diferencias Finitas en el Dominio del Tiempo*, en donde numéricamente se calcula el campo electromagnético tomando en cuenta las nanoestructuras diseñadas así como el material involucrado en la simulación.

El tipo de modelado que se genera a partir de la simulación computacional es una herramienta poderosa para la visualización en el tiempo del comportamiento del campo electromagnético a través de la nanoestructura.

Vale la pena mencionar que con una serie de sistemas planteados se pretende descubrir comportamientos en fenómenos a nanoescala y al mismo tiempo ilustrar la manera de realizar simulaciones por medio de Phoxonics. Se pretende lograr una analogía de laboratorio virtual en donde cada uno de los sistemas se convierte en experimento en la medida que se introducen nuevos parámetros y variables. Es posible combinar diferentes factores y geometrías que afecten a los análisis computacionales obteniendo múltiples resultados en forma de datos, gráficas y animaciones.

Por último, se estará introduciendo una herramienta de software que se ha creado llamada Phoxonics la cual cuenta con características que se consideran valiosas para el recorrido del vasto e interesante mundo de la nanotecnología computacional.

Yohan J. Rodríguez Viveros
Universidad de Sonora

Agradecimientos

Quiero agradecer a mis padres y mis hermanos, a mis hijos, a mi esposa y su familia. Gracias por apoyarme en este proyecto tan importante para mi. Su motivación y sus palabras me han hecho valorar lo afortunado que soy de tenerlos a mi lado.

Gracias a mi asesor Dr. Jesús Manzanares Martínez. Por compartir su conocimiento conmigo y enseñarme a tener una nueva visión de los problemas planteados. Por su paciencia, persistencia y motivación en ayudarme a alcanzar las metas deseadas. Por inculcarme el valor de plantearse las cosas de manera novedosa. Por ser mi guía a lo largo de todo el trayecto.

Gracias a mis compañeros de investigación Damian Moctezuma e Ivan Ham. Por estar al pie del cañón siempre que había que tratar temas nuevos. Por su compañerismo al tratarse de plantear problemas de su especialidad y por ayudar a abrir la mente con pensamientos de otras disciplinas.

Un agradecimiento a la Universidad de Sonora y al Conacyt por todo el apoyo brindado.

Sin ustedes este proyecto no hubiera sido posible.
¡Muchas gracias!

Tabla de Contenido

Lista de Figuras

Listas de Código

1 Introducción

En este capítulo se exponen los antecedentes relacionados a esta tesis. Se presentan las múltiples disciplinas relacionadas con la nanotecnología y el cómputo. Se establece la manera en que las disciplinas interaccionan para lograr describir y descubrir los fenómenos de los sistemas que se plantean.

1.1. La Nanotecnología y la Nanociencia

La nanotecnología y la nanociencia se han convertido en los principales campos hacia una revolución tecnológica en el nuevo milenio. La nanotecnología consiste en la investigación a través de múltiples disciplinas como la ingeniería, la física, la química, la biología y la medicina.

La nanotecnología se refiere a la ciencia de la materia que es menor a $100nm$ [1]. Por lo tanto, la capacidad de manipular la materia en la escala del nanómetro ayuda a los científicos a crear pequeños dispositivos del tamaño de algunos átomos para diversas aplicaciones. La nanotecnología se aplica en los campos de la información y la biotecnología, aeroespacial, energía y medio ambiente, en la mejora de la administración de fármacos identificando las células cancerosas en el área de medicina, etc. [2].

El uso de la nanotecnología trae muchos beneficios a la industria. La ciencia y la ingeniería en la escala del nanómetro ofrecen la posibilidad de avances revolucionarios en ambos campos y pueden tener un impacto en nuestra economía y en la sociedad que es comparable en escala y alcance a la electrónica de transistores [3].

En su nivel básico, la nanociencia es el estudio de los fenómenos y propiedades de los materiales que se producen en escalas de longitud del nanómetro [4]. De manera similar, la nanotecnología es la aplicación de los métodos de la nanociencia y la ingeniería para producir nuevos materiales y dispositivos [5]. Comúnmente se utiliza el término nanotecnología para referirse a ambas disciplinas.

1.2. Las Estructuras Nanométricas

Las estructuras a escalas nanométricas son interesantes ya que generalmente en la región del nanómetro casi todas las propiedades físicas y químicas de los sistemas se vuelven dependientes

del tamaño. Por ejemplo, aunque el color de una pieza de oro se mantiene dorado al reducirla de centímetros a micras, el color cambia sustancialmente en el régimen de los nanómetros. Del mismo modo las temperaturas de fusión de sistemas pequeños cambia a medida que entran en la región del nanómetro. Dado que las propiedades en la escala del nanómetro son dependientes del tamaño, la ciencia y la ingeniería a nanoescala ofrecen una visión totalmente nueva de diseño para el desarrollo de materiales avanzados y nuevos dispositivos.

La revolución de la nanociencia ha creado una necesidad urgente de una comprensión cuantitativa de la materia a escalas nanométricas mediante el modelado y la simulación ya que la ausencia de modelos cuantitativos limita cada vez más el progreso en este campo. Aunque existen técnicas y métodos de modelado conocidos, los nuevos conocimientos provienen de la aplicación de estas técnicas y métodos en el campo de la nanociencia. Los avances en la tecnología informática y el aumento de las capacidades computacionales han hecho posible el modelado y simulación de sistemas cada vez más complejos [3].

El interés en las propiedades ópticas de nanoestructuras metálicas ha ido aumentando continuamente en la medida en que las técnicas experimentales para su fabricación e investigación se vuelven más sofisticadas [6]. Uno de los principales factores de este interés es su potencial utilidad en aplicaciones optoelectrónicas y fotónica [6, 7]. Sin embargo, también puede haber problemas fundamentales interesantes a considerar, en particular, a medida que se aproximan escalas de longitud muy pequeñas, aproximadamente menos de $10nm$ [8]. En este límite, los efectos de la mecánica cuántica puede conducir a propiedades ópticas inusuales en relación con las predicciones basadas en la electrodinámica clásica [8].

1.3. El Desarrollo de la Nanotecnología

Se puede decir que las áreas de investigación de mayor crecimiento en los últimos 10 años que pertenecen a la física, química e ingeniería son las ciencias de materiales y la nanotecnología. Si bien es cierto que las ciencias de materiales siempre han sido importantes, también se puede afirmar que hasta hace poco estaban de alguna manera limitadas en alcance. Antes de la revolución de la nanotecnología, toda la investigación en ciencias de materiales fue básicamente dominada por la física y la ingeniería. El mayor esfuerzo detrás de esta investigación fueron los intentos en el área de electrónica e informática por miniaturizar transistores y procesadores electrónicos. En esencia todo fue una estrategia en donde a partir de un dispositivo macroscópico se intentaba hacerlo más pequeño y más pequeño cada vez.

La nanotecnología introdujo un cambio absoluto en esta forma de pensar. La nueva rama de ciencias de materiales a nanoescala está enfocada en una estrategia de abajo hacia arriba. Hoy en día, se quiere manipular átomos ó moléculas de tal manera de formar nuevas nanoestructuras artificiales con propiedades definidas. En la actualidad, la nanotecnología va desde la nanoelectrónica hasta aplicaciones biomédicas, y la importancia de este campo no puede por ningún motivo ser subestimada.

La nanotecnología ofrece posibilidades ilimitadas en el proceso de avance de muchas ciencias físicas y de ingeniería. También, ofrece posibilidades sin precedentes para el desarrollo de nuevas tecnologías. Una amplia variedad de nanomateriales son utilizados en la actualidad en ingeniería, industria farmacéutica, productos biomédicos, así como en otras industrias. Mientras que los materiales a nanoescala poseen propiedades físico-químicas novedosas y únicas, no son fáciles de estudiar. Por lo tanto, el estudio de los nanomateriales ha generado una intensa curiosidad científica, atrayendo mucha atención en los últimos años [9].

1.4. La Nanotecnología Computacional

El método FDTD y sus avances apoyan el progreso de las aplicaciones de la nanotecnología computacional. El método FDTD es una herramienta esencial en el modelado de nanoestructuras debido a sus características de extrema flexibilidad e implementación relativamente sencilla. Por ejemplo, existe un interés en el estudio de aplicaciones nanofotónicas y su optimización mediante el método FDTD. Además, se han reportado estudios de optimización en el campo electromagnético en nanocilindros de plata ordenados en geometrías de triángulo [10].

Los retos del método FDTD siguen atrayendo investigadores de todo el mundo y han dado lugar a muchos de los nuevos descubrimientos. No hay duda de que el método FDTD brinda luz hacia otras disciplinas, como la ingeniería, la física, la química, la biología, la medicina y la industria aeroespacial. Por lo tanto, el estudio del método FDTD computacional en la escala del nanómetro es vital debido al rápido avance de la nanotecnología y la nanociencia [10].

1.5. La Nanofotónica Computacional

Este subcampo ha surgido en recientes años. Desempeña un papel cada vez más importante a medida que las dimensiones de los dispositivos fotónicos se hacen más pequeñas. Las nanoestructuras son generalmente consideradas por tener tamaños del orden de la longitud de onda de la luz, es decir, si la guía transmite un solo modo, entonces $d = \lambda$, donde d es el diámetro y lambda es la longitud de onda de la luz. En esas longitudes de onda, múltiples efectos de dispersión y de campo cercano tienen una profunda influencia en la propagación de la luz y en la interacción luz-materia. A su vez, esto conduce a nuevos regímenes para la investigación básica, así como nuevas aplicaciones en diversas disciplinas.

Por ejemplo, la relación de dispersión modificada de los cristales fotónicos y de las fibras de cristal fotónico conduce a nuevos efectos de propagación de ondas no lineales tales como los cambios de solitones gigantes con aplicaciones en telecomunicaciones, metrología y diagnóstico médico. Debido a la compleja naturaleza de los procesos de interferencia de onda y de interacción, los estudios experimentales dependen fuertemente de la orientación teórica tanto para el diseño de tales sistemas, así como para la interpretación de las mediciones. En casi todos los casos, una descripción

teórica cuantitativa de tales sistemas tiene que estar basada en técnicas computacionales avanzadas que resuelven numéricamente grandes ecuaciones diferenciales parciales acopladas, lineales o no lineales.

La investigación de las estructuras de cristal fotónico aplicadas a nuevos componentes en la comunicación óptica, ha crecido exponencialmente desde que se ha hecho posible crear estructuras en la escala del nanómetro. La nanotecnología ofrece un potencial para circuitos ópticos integrados más eficientes a un costo menor, desde elementos pasivos como filtros y ecualizadores hasta funciones activas tales como conmutación óptica, interconexiones e incluso nuevos láseres. Además, la tecnología puede ofrecer nuevas funcionalidades y reducir los costos generales de comunicación óptica gracias a dispositivos más pequeños, mayor ancho de banda y las características de bajas pérdidas de dispositivos que eliminan la necesidad de amplificación. El objetivo final sería la creación de estructuras fotónicas en tres dimensiones que puedan eventualmente conducir a la computación totalmente óptica [11].

Una visión muy reciente en el modelado de la fotónica incluyendo importantes métodos computacionales fue escrito por Obayya [12]. Según Joannopoulos *et al.* [13] en un sentido amplio existen tres categorías de problemas en fotónica computacional: eigenproblemas en el dominio de la frecuencia: consiste en encontrar la estructura de bandas y los campos asociados expresando el problema como un eigenproblema de matriz finita. Respuestas en el dominio de la frecuencia: dada una distribución de corriente en una frecuencia fija, consiste en encontrar los campos resultantes expresando el problema como una ecuación de matriz finita. Y simulaciones en el dominio del tiempo, estos problemas se discuten en [13, 14, 15].

1.6. ¿Por qué Estudiar la Nanoescala?

La complejidad y variedad de aplicaciones en la nanoescala son tan grandes o posiblemente mayores que en la macroescala. Las propiedades de los materiales se ven fuertemente afectadas por su estructura a nanoescala. Durante las dos últimas décadas, la humanidad ha estado inventando y adquiriendo gradualmente medios para caracterizar y manipular dicha estructura. Muchos efectos notables, fenómenos físicos, materiales y dispositivos han sido descubiertos o desarrollados: nanocompuestos, nanotubos de carbono, nanocables y nanopuntos, nanopartículas de diferentes tipos, cristales fotónicos etc. En un nivel más fundamental, la investigación en física a nanoescala puede proporcionar pistas sobre los más profundos misterios de la naturaleza.

1.7. ¿Por qué Modelos Especiales Para la Nanoescala?

En primer lugar, una observación general. Un modelo de simulación consiste en una formulación física y matemática del problema en cuestión y un método computacional. La formulación nos dice lo que debemos resolver y el método computacional nos dice cómo resolverlo. Frecuentemente más

de una formulación es posible, y casi siempre múltiples técnicas computacionales están disponibles; por lo tanto existen numerosas combinaciones de formulaciones y métodos. Idealmente, uno se esfuerza por encontrar las mejores combinaciones en términos de eficiencia, precisión, robustez, simplicidad y así sucesivamente.

1.8. ¿Cómo Perfeccionar las Herramientas Computacionales?

La simulación computacional no es una ciencia exacta. Si lo fuera, se podría simplemente establecer un nivel deseado de precisión en la solución numérica y demostrar que un determinado método logra ese nivel con el número mínimo de operaciones. La realidad es, por supuesto, mucho más complicada.

En primer lugar, hay muchas medidas posibles de precisión y muchas medidas posibles de costo (teniendo en cuenta el tiempo humano necesario para el desarrollo de algoritmos y software puede que éste sea más valioso que el tiempo de CPU). La precisión y el costo dependen directamente de la clase y subclase de los problemas a resolver. Por ejemplo, la solución numérica se vuelve sustancialmente más complicada si existen discontinuidades o singularidades en bordes y esquinas del campo que requieren ser representadas con precisión.

En segundo lugar, por lo general, es casi imposible garantizar, a nivel matemático, que la solución numérica tiene una precisión exacta.

En tercer lugar, hay errores de modelado. Aproximaciones hechas en la formulación del problema físico; estos errores son motivo especial de preocupación a nivel nanométrico, donde es muy difícil la verificación experimental directa y precisa de los supuestos hechos.

En cuarto lugar, una serie de otros temas. Desde la implementación algorítmica del método elegido para los errores de redondeo, que son bastante difíciles de tener en cuenta. La paralelización del algoritmo y el código computacional es otro tema complicado. Teniendo todo esto en mente, la simulación computacional resulta ser parcialmente un arte. Siempre hay más de una manera de resolver un problema numérico y, con el tiempo y los recursos suficientes, un enfoque razonable es probable que produzca un resultado eventualmente [16].

1.9. Objetivos de la Investigación

Este trabajo tiene como objetivos los siguientes:

- Estudiar los fenómenos físicos relacionados con la propagación de ondas electromagnéticas a través de estructuras en la escala del nanómetro mediante el Método de Diferencias Finitas en el Dominio del Tiempo.

- Desarrollar una herramienta computacional que utilice el método FDTD y que permita diseñar estructuras en la escala del nanómetro para estudiar la propagación de ondas electromagnéticas a través de ellas.
- Fomentar la utilización del método FDTD en el estudio de fenómenos de propagación de ondas electromagnéticas en la escala del nanómetro.
- Publicar resultados obtenidos en forma de artículos científicos.

1.10. Pregunta e Hipótesis

1.10.1. Pregunta

¿Es posible investigar fenómenos físicos relacionados con la propagación de ondas electromagnéticas en estructuras a nivel nanómetro mediante simulación computacional?

1.10.2. Hipótesis

Se espera que con el desarrollo de una herramienta computacional que implemente el método FDTD sea posible investigar fenómenos físicos relacionados con la propagación de ondas electromagnéticas en estructuras a nivel nanómetro

1.11. Justificación

1.11.1. Descripción del Problema

Existen fenómenos físicos relacionados con la propagación del campo electromagnético a través de estructuras a nivel nanómetro. Este tipo de fenómenos puede ser estudiado mediante herramientas como la física y la matemática. En los últimos años el poder del cálculo computacional se ha incrementado exponencialmente, con esto, se abre la posibilidad de utilizar este poder de cálculo computacional para investigar dichos fenómenos mediante el método FDTD. A partir de lo anterior, ha surgido la necesidad de desarrollar técnicas computacionales para investigación de fenómenos de propagación de ondas electromagnéticas en estructuras a nivel nanómetro.

1.11.2. Objetivo General

Se pretende utilizar la simulación computacional como herramienta en la investigación de fenómenos relacionados con la propagación de ondas electromagnéticas en estructuras a nivel nanómetro.

1.11.3. Objetivo Específico

Se pretende escribir artículos científicos como resultado de la investigación de fenómenos relacionados con la propagación de ondas electromagnéticas a nivel nanómetro.

1.12. Referencias

[1] Mark A. Ratner and Daniel Ratner. *Nanotechnology: A Gentle Introduction to the Next Big Idea*. Prentice Hall, 2002.

[2] T. Pradeep. *Nano: The Essentials*. McGraw-Hill Education, 2008.

[3] Sarhan M. Musa. *Computational Nanotechnology: Modeling and Applications with MATLAB (Nano and Energy)*. CRC Press, 2011.

[4] Massimiliano Ventra, Stephane Evoy, and James R. Heflin. *Introduction to Nanoscale Science and Technology (Nanostructure Science and Technology)*. Springer, 2004.

[5] Louis Theodore. *Nanotechnology: Basic Calculations for Engineers and Scientists*. Wiley-Interscience, 2005.

[6] Katherine A. Willets and Richard P. Van Duyne. Localized surface plasmon resonance spectroscopy and sensing. *Annual Review of Physical Chemistry*, 58(1):267–297, may 2007.

[7] E. Ozbay. Plasmonics: Merging photonics and electronics at nanoscale dimensions. *Science*, 311(5758):189–193, jan 2006.

[8] Uwe Kreibig and Michael Vollmer. *Optical Properties of Metal Clusters (Springer Series in Materials Science)*. Springer, 1995.

[9] Sarhan M. Musa. *Computational Finite Element Methods in Nanotechnology*. CRC Press, 2012.

[10] Sarhan M. Musa. *Computational Nanotechnology Using Finite Difference Time Domain*. CRC Press, 2013.

[11] Marek S. Wartak. *Computational Photonics: An Introduction with MATLAB*. Cambridge University Press, 2013.

[12] Salah Obayya. *Computational Photonics*. Wiley, 2010.

[13] John D. Joannopoulos, Steven G. Johnson, Joshua N. Winn, and Robert D. Meade. *Photonic Crystals: Molding the Flow of Light*. Princeton University Press, 2008.

[14] Jian-Ming Jin. *Theory and Computation of Electromagnetic Fields*. Wiley-IEEE Press, 2010.

[15] Allen Taflove and Susan C. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method, Third Edition*. Artech House, 2005.

[16] Igor Tsukerman. *Computational Methods for Nanoscale Applications: Particles, Plasmons and Waves (Nanostructure Science and Technology)*. Springer, 2007.

2 Teoría

Puede decirse que el electromagnetismo moderno está basado en una invención y dos descubrimientos, realizados en el primer tercio del siglo XIX [1, 2]. La invención es la construcción de una fuente de corriente eléctrica continua, la pila eléctrica, llevada a cabo por Volta hacia 1800. Los dos descubrimientos son la demostración de los efectos magnéticos producidos por corrientes eléctricas realizada por Oersted y Ampére en 1820 y la generación de corriente eléctrica a partir de campos magnéticos obtenida por Faraday en 1831 [1, 2]. Los trabajos de Oersted y Ampére permitieron sentar las bases experimentales y matemáticas del electromagnetismo, mientras que Faraday es el responsable, además, de la introducción del concepto de "campo" para describir las fuerzas eléctricas y magnéticas [1, 2, 3, 4, 5, 6], idea revolucionaria en su tiempo pues suponía alejarse de la descripción mecanicista de los fenómenos naturales al más puro estilo newtoniano, es decir, mediante "acciones a distancia" [2] sin intermediación de ningún medio. Con estas tres contribuciones se habían puesto los pilares del moderno electromagnetismo, cerrado por la aportación de James Clerk Maxwell, ya en el ultimo tercio del siglo XIX [4, 5]. Con Oersted y Ampére primero, y los trabajos de Faraday después, empieza a gestarse la síntesis electromagnética de Maxwell [2].

En este capítulo se explora la teoría relacionada con el electromagnetismo y su desarrollo a partir de las ecuaciones de Maxwell, se plantea la manera de llegar a la ecuación de onda y cómo utilizarla para describir y resolver sistemas analíticamente.

2.1. Las Ecuaciones del Campo Electromagnético

Primero se presentan las ecuaciones de Maxwell en el sistema internacional de unidades Metro-Kilogramo-Segundo (MKS) y se realiza una normalización para llegar a las ecuaciones en el sistema de unidades Centímetro-Gramo-Segundo (CGS), con el fin de facilitar el manejo de unidades y hacer más fácil la programación de las ecuaciones al momento de la discretización. Las ecuaciones de Maxwell en MKS son

$$\nabla \cdot \mathbf{d}(\mathbf{x}, t) = P(\mathbf{x}, t), \tag{2.1}$$

$$\nabla \cdot \mathbf{b}(\mathbf{x}, t) = 0, \tag{2.2}$$

$$\nabla \times \mathbf{h}(\mathbf{x},t) = \mathbf{j}(\mathbf{x},t) + \frac{\partial}{\partial t}\mathbf{d}(\mathbf{x},t), \tag{2.3}$$

$$\nabla \times \mathbf{e}(\mathbf{x},t) = -\frac{\partial}{\partial t}\mathbf{b}(\mathbf{x},t), \tag{2.4}$$

donde $\mathbf{d}$, $\mathbf{b}$, $\mathbf{h}$ y $\mathbf{e}$ son los campos en MKS. $P(\mathbf{x},t)$ y $\mathbf{j}(\mathbf{x},t)$ son las densidades de carga y corriente en MKS respectivamente. Se proponen las relaciones

$$\mathbf{D}(\mathbf{x},t) = \sqrt{\frac{4\pi}{\varepsilon_0}}\ \mathbf{d}(\mathbf{x},t), \tag{2.5}$$

$$\rho(\mathbf{x},t) = \frac{1}{\sqrt{4\pi\varepsilon_0}}\ P(\mathbf{x},t), \tag{2.6}$$

$$\mathbf{B}(\mathbf{x},t) = \sqrt{\frac{4\pi}{\mu_0}}\ \mathbf{b}(\mathbf{x},t), \tag{2.7}$$

$$\mathbf{H}(\mathbf{x},t) = \sqrt{4\pi\mu_0}\ \mathbf{h}(\mathbf{x},t), \tag{2.8}$$

$$\mathbf{E}(\mathbf{x},t) = \sqrt{4\pi\varepsilon_0}\ \mathbf{e}(\mathbf{x},t), \tag{2.9}$$

$$\mathbf{J}(\mathbf{x},t) = \frac{1}{\sqrt{4\pi\varepsilon_0}}\ \mathbf{j}(\mathbf{x},t), \tag{2.10}$$

donde los campos en CGS corresponden a

- $\mathbf{E}(\mathbf{x},t)$ es el campo eléctrico,
- $\mathbf{B}(\mathbf{x},t)$ es la inducción magnética,
- $\mathbf{H}(\mathbf{x},t)$ es el campo magnético y
- $\mathbf{D}(\mathbf{x},t)$ es el desplazamiento eléctrico.

Las fuentes de campo electromagnético son

- $\mathbf{J}(\mathbf{x},t)$ que es la densidad de corriente y
- $\rho(\mathbf{x},t)$ que es la densidad de carga.

Entonces se puede plantear

$$\nabla \cdot \mathbf{D}(\mathbf{x},t) = 4\pi\rho(\mathbf{x},t), \tag{2.11}$$

$$\nabla \cdot \mathbf{B}(\mathbf{x},t) = 0, \tag{2.12}$$

$$\nabla \times \mathbf{H}(\mathbf{x},t) = \frac{4\pi}{c}\mathbf{J}(\mathbf{x},t) + \frac{1}{c}\frac{\partial}{\partial t}\mathbf{D}(\mathbf{x},t), \tag{2.13}$$

$$\nabla \times \mathbf{E}(\mathbf{x},t) = -\frac{1}{c}\frac{\partial}{\partial t}\mathbf{B}(\mathbf{x},t). \tag{2.14}$$

Así se puede pensar que éstas son las ecuaciones MKS *normalizadas* donde $c = \frac{1}{\sqrt{\varepsilon_0\mu_0}}$ es la velocidad de la luz, o bien, las ecuaciones de Maxwell en CGS [7]. Las ecuaciones de Maxwell se complementan con las relaciones constitutivas en el dominio de la frecuencia

$$\mathbf{D}(\mathbf{x},\omega) = \varepsilon(\mathbf{x},\omega)\mathbf{E}(\mathbf{x},\omega), \tag{2.15}$$

$$\mathbf{B}(\mathbf{x},\omega) = \mu(\mathbf{x},\omega)\mathbf{H}(\mathbf{x},\omega), \tag{2.16}$$

donde $\varepsilon(\mathbf{x},\omega)$ y $\mu(\mathbf{x},\omega)$ son la función dieléctrica y permeabilidad magnética, respectivamente. La relación entre los campos en el dominio del tiempo y la frecuencia se establece a partir de la transformada de Fourier en la forma

$$\mathbf{A}(\mathbf{x},t) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty}\mathbf{A}(\mathbf{x},\omega)e^{-i\omega t}d\omega, \tag{2.17}$$

$$\mathbf{A}(\mathbf{x},\omega) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty}\mathbf{A}(\mathbf{x},t)e^{i\omega t}dt. \tag{2.18}$$

2.1.1. Medio Dieléctrico Homogéneo y no Dispersivo

Se considera un medio dieléctrico homogéneo y no dispersivo eliminando la frecuencia de la forma

$$\varepsilon(\mathbf{x},\omega) = \varepsilon, \tag{2.19}$$

$$\mu(\mathbf{x},\omega) = \mu. \tag{2.20}$$

Además se considera que no existen fuentes de carga, ni de corriente

$$\rho(\mathbf{x},t) = 0, \tag{2.21}$$

$$\mathbf{J}(\mathbf{x},t) = 0. \tag{2.22}$$

Las ecuaciones de Maxwell se reducen a

$$\nabla \times \mathbf{E}(\mathbf{x},t) = -\frac{\mu}{c}\frac{\partial}{\partial t}\mathbf{H}(\mathbf{x},t), \tag{2.23}$$

$$\nabla \times \mathbf{H}(\mathbf{x},t) = \frac{\varepsilon}{c}\frac{\partial}{\partial t}\mathbf{E}(\mathbf{x},t), \tag{2.24}$$

$$\nabla \cdot \mathbf{E}(\mathbf{x},t) = 0, \tag{2.25}$$

$$\nabla \cdot \mathbf{H}(\mathbf{x},t) = 0. \tag{2.26}$$

Para llegar a la ecuación de onda se aplica el rotacional en ambos lados de la ecuación de Faraday 2.23 y se aplica la siguiente identidad vectorial

$$\nabla \times [\nabla \times \mathbf{E}(\mathbf{x},t)] = \nabla [\nabla \cdot \mathbf{E}(\mathbf{x},t)] - \nabla^2 \mathbf{E}(\mathbf{x},t) \tag{2.27}$$

Aplicando la identidad vectorial y sustituyendo con las ecuaciones de Maxwell, es posible llegar a la ecuación de onda para el campo eléctrico

$$\nabla^2 \mathbf{E}(\mathbf{x},t) = \frac{\mu\varepsilon}{c^2}\frac{\partial^2}{\partial t^2}\mathbf{E}(\mathbf{x},t) \tag{2.28}$$

2.1.2. La Ecuación de Onda en Una Dimensión

Ahora se plantea una polarización del campo eléctrico de la forma

$$\mathbf{E}(\mathbf{x},t) = \hat{k}E_z(y,t) \tag{2.29}$$

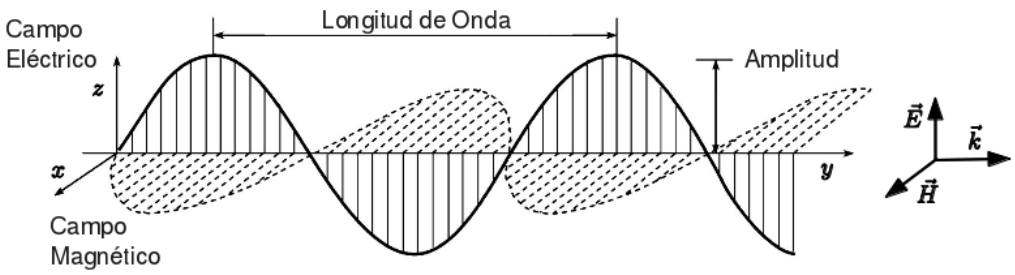

Figura 2.1: Onda electromagnética

Ahora se puede escribir la ecuación de onda (2.28) en una dimensión

$$\frac{\partial^2}{\partial y^2}E_z(y,t) = \frac{\mu\varepsilon}{c^2}\frac{\partial^2}{\partial t^2}E_z(y,t) \tag{2.30}$$

Para resolver esta ecuación se plantea una separación de variables en la forma

$$E_z(y,t) = Y(y)T(t) \tag{2.31}$$

Sustituyendo la ecuación (2.31) en la ecuación (2.30) se tiene

$$\frac{1}{Y(y)}\frac{\partial^2}{\partial y^2}Y(y) = \frac{\mu\varepsilon}{c^2}\frac{1}{T(t)}\frac{\partial^2}{\partial t^2}T(t) \tag{2.32}$$

De acuerdo al método de separación de variables, cada una de las ecuaciones diferenciales de segundo orden se puede igualar a una constante arbitraria llamada constante de separación, la cual debe ser analizada en sus posibilidades de signo. Para la ecuación en y se tiene

$$\frac{1}{Y(y)}\frac{\partial^2}{\partial y^2}Y(y) = \pm k_y^2 \tag{2.33}$$

donde k_y es una constante arbitraria que puede ser positiva o negativa. Para la ecuación en t se tiene

$$\frac{1}{T(t)}\frac{\partial^2}{\partial t^2}T(t) = \pm\omega^2 \tag{2.34}$$

donde ω es una constante arbitraria que también puede ser positiva o negativa

Soluciones Para $Y(y)$

- Para el caso $-k_y^2$ se tiene la ecuación diferencial

$$\frac{\partial^2}{\partial y^2}Y(y) = -k_y^2 Y(y) \tag{2.35}$$

Se propone la solución $e^{ik_y y}$ la cual cumple con el criterio que se busca ya que

$$Y(y) = e^{ik_y y}, \tag{2.36}$$

$$\frac{\partial}{\partial y}Y(y) = ik_y e^{ik_y y}, \tag{2.37}$$

$$\frac{\partial^2}{\partial y^2}Y(y) = -k_y^2 Y(y). \tag{2.38}$$

Se considera la identidad de Euler $e^{ix} = \cos(x) + i\sin(x)$, entonces se puede decir que se tienen dos soluciones armónicas linealmente independientes que se ilustran en la Figura 2.2 y que se escriben en la forma

$$Y(y) = Y_1 \cos(k_y y) + Y_2 \sin(k_y y) \tag{2.39}$$

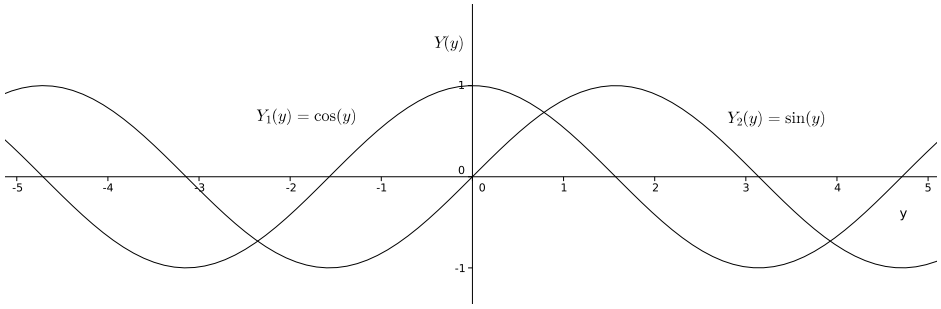

Figura 2.2: Solución armónica para $Y(y)$

- Para el caso $+k_y^2$ se tiene la ecuación diferencial

$$\frac{\partial^2}{\partial y^2} Y(y) = +k_y^2 Y(y) \tag{2.40}$$

Se propone la solución $e^{k_y y}$ la cual cumple con el criterio que se busca ya que

$$Y(y) = e^{k_y y}, \tag{2.41}$$

$$\frac{\partial}{\partial y} Y(y) = k_y e^{k_y y}, \tag{2.42}$$

$$\frac{\partial^2}{\partial y^2} Y(y) = k_y^2 Y(y). \tag{2.43}$$

Considerando que la solución $e^{-k_y y}$ arroja el mismo resultado, entonces se puede decir que se tienen dos soluciones linealmente independientes que determinan campos crecientes o decrecientes en la dirección y en la forma

$$Y(y) = Y_3 e^{-k_y y} + Y_4 e^{+k_y y} \tag{2.44}$$

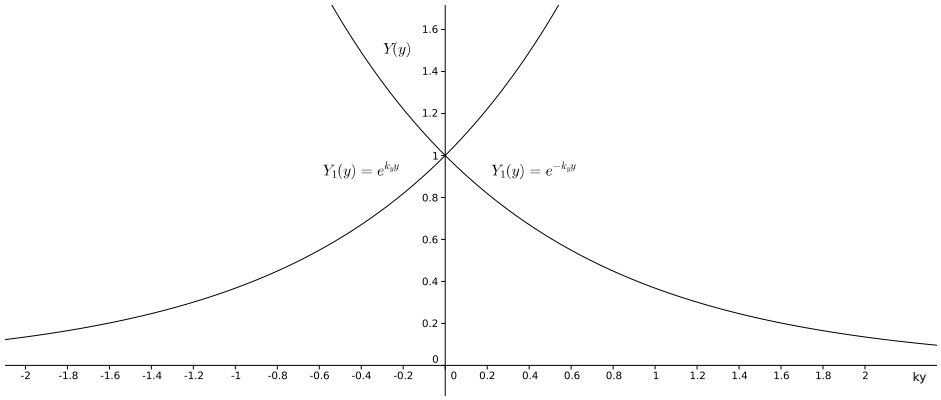

Figura 2.3: Solución creciente y decadente para $Y(y)$

2.1.3. La Ecuación de Onda en Dos Dimensiones

Ahora se plantea una polarización del campo eléctrico de la forma

$$\mathbf{E}(\mathbf{x}, t) = \hat{k} E_z(x, y, t) \tag{2.45}$$

Se tiene una ecuación espacial y temporal de la forma

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) E_z(x, y, t) = \frac{\mu\varepsilon}{c^2} \frac{\partial^2}{\partial t^2} E_z(x, y, t) \tag{2.46}$$

Se considera que

$$E_z(x, y, t) = X(x)Y(y)T(t) \tag{2.47}$$

Así, mediante la separación de variables se puede escribir

$$\frac{1}{X(x)} \frac{\partial^2}{\partial x^2} X(x) + \frac{1}{Y(y)} \frac{\partial^2}{\partial y^2} Y(y) = \frac{\mu\varepsilon}{c^2} \frac{1}{T(t)} \frac{\partial^2}{\partial t^2} T(t) \tag{2.48}$$

Para el segundo término del lado izquierdo de la ecuación 2.48 se propone una solución armónica de la forma

$$Y(y) = Y^+ e^{ik_y y} + Y^- e^{-ik_y y} \tag{2.49}$$

la cual cumple con el criterio que se busca ya que

$$\frac{\partial}{\partial y} Y(y) = (ik_y) Y^+ e^{ik_y y} + (-ik_y) Y^- e^{-ik_y y}, \tag{2.50}$$

$$\frac{\partial^2}{\partial y^2} Y(y) = -k_y^2 Y(y). \tag{2.51}$$

De esta forma se puede escribir

$$\frac{1}{Y(y)}\frac{\partial^2}{\partial y^2}Y(y) = -k_y^2 \tag{2.52}$$

De la misma forma para el lado derecho de la ecuación 2.48 se propone una solución armónica de la forma

$$T(t) = T^+e^{i\omega t} + T^-e^{-i\omega t} \tag{2.53}$$

la cual cumple con el criterio que se busca ya que

$$\frac{\partial}{\partial t}T(t) = (i\omega)T^+e^{i\omega t} + (-i\omega)T^-e^{-i\omega t}, \tag{2.54}$$

$$\frac{\partial^2}{\partial t^2}T(t) = -\omega^2 T(t). \tag{2.55}$$

De esta forma se puede escribir

$$\frac{1}{T(t)}\frac{\partial^2}{\partial t^2}T(t) = -\omega^2 \tag{2.56}$$

De esta manera se tiene a la ecuación 2.48 en la forma

$$\frac{1}{X(x)}\frac{\partial^2}{\partial x^2}X(x) - k_y^2 = -k^2 \tag{2.57}$$

donde

$$k^2 = \mu\varepsilon\frac{\omega^2}{c^2} \tag{2.58}$$

Entonces tenemos

$$\frac{1}{X(x)}\frac{\partial^2}{\partial x^2}X(x) = -k^2 + k_y^2 \tag{2.59}$$

Ahora se procede a identificar dos casos, $k > k_y$ y $k < k_y$

Caso $k > k_y$

En este caso se tiene que $-k^2 + k_y^2 < 0$, así se propone

$$X(x) = X^+e^{ik_x x} + X^-e^{-ik_x x}, \tag{2.60}$$

$$\frac{\partial}{\partial x}X(x) = (ik_x)X^+e^{ik_x x} + (-ik_x)X^-e^{-ik_x x}, \tag{2.61}$$

$$\frac{\partial^2}{\partial x^2}X(x) = -k_x^2 X(x), \tag{2.62}$$

$$\frac{1}{X(x)}\frac{\partial^2}{\partial x^2}X(x) = -k_x^2, \tag{2.63}$$

de esta manera

$$k_x = \sqrt{k^2 - k_y^2} \tag{2.64}$$

Si se considera el caso en el que $X^+ = X^-$, se tiene una solución de la forma

$$X(x) = X^+ \cos(k_x x) \tag{2.65}$$

Ahora bien, en el caso en el que $X^+ = -X^-$ se tiene una solución de la forma

$$X(x) = X^+ \sin(k_x x) \tag{2.66}$$

Caso $k < k_y$

En este caso se tiene que $-k^2 + k_y^2 > 0$, así se propone

$$X(x) = X^+ e^{k_x x} + X^- e^{-k_x x}, \tag{2.67}$$

$$\frac{\partial}{\partial x}X(x) = k_x X^+ e^{k_x x} - (-k_x) X^- e^{-k_x x}, \tag{2.68}$$

$$\frac{\partial^2}{\partial x^2}X(x) = +k_x^2 X(x), \tag{2.69}$$

$$\frac{1}{X(x)}\frac{\partial^2}{\partial x^2}X(x) = +k_x^2. \tag{2.70}$$

De esta manera se plantea que

$$k_x = \sqrt{k_y^2 - k^2} \tag{2.71}$$

De esta forma se tiene que si

$$k > k_y \implies k_x = \sqrt{k^2 - k_y^2} \quad \text{Se tienen ondas armónicas} \tag{2.72}$$

$$k < k_y \implies k_x = \sqrt{k_y^2 - k^2} \quad \text{Se tienen ondas decadentes} \tag{2.73}$$

El valor límite de estas dos regiones es

$$k_y^2 - k^2 = 0, \tag{2.74}$$

$$k_y^2 = \mu\varepsilon\frac{\omega^2}{c^2}. \tag{2.75}$$

A esto se le llama la línea de luz, ya que define la división entre las regiones

$$\omega = \frac{c}{\sqrt{\mu\varepsilon}}k_y \tag{2.76}$$

De esta forma se pueden visualizar las regiones oscilantes y decadentes para un medio homogéneo.

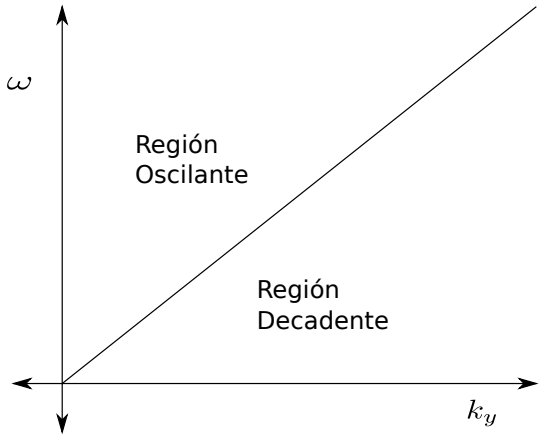

Figura 2.4: Soluciones oscilantes / decadentes para un medio homogéneo

2.2. Relación de Dispersión de una Guía Plana

Las guías de onda se basan en el confinamiento de la luz, efecto que se logra mediante el uso de dos medios con índice de refracción diferente. El medio con índice de refracción mayor (núcleo) se embebe en el medio con índice de refracción menor (revestimiento). La luz queda confinada en el medio del núcleo debido al principio de reflexión total interna [8].

El principio de la reflexión total interna está basado en la propagación de la luz en una guía de ondas. Cuando la luz pasa de un medio más denso a otro menos denso puede ser reflejada internamente. Cuando el ángulo de incidencia del rayo luminoso es mayor que el valor del llamado ángulo crítico la luz se refleja internamente totalmente. La reflexión total interna se rige por dos factores: los índices de refracción de los dos medios, n_a y n_b y el ángulo crítico θ_c.

$$\theta_c = \arcsin(n_b/n_a) \tag{2.77}$$

Existen varios tipos de geometrías de ondas como las cilíndricas utilizadas en fibras ópticas. En este caso se plantea una guía de onda plana con una función dieléctrica con una dependencia espacial de la forma

$$\varepsilon(\mathbf{x}) = \varepsilon(x) \tag{2.78}$$

donde $\varepsilon(x)$ se ilustra en la Figura 2.5 y tiene la forma

$$\varepsilon(x) = \begin{cases} \varepsilon_b & x < -d/2 \\ \varepsilon_a & -d/2 < x < d/2 \\ \varepsilon_b & x > d/2 \end{cases} \tag{2.79}$$

La placa central tiene una *alta* función dieléctrica ε_a mientras que el medio externo a esa placa tiene una *baja* función dieléctrica ε_b. Para tener ondas guiadas se requiere que $\varepsilon_a > \varepsilon_b$. A continuación se muestra una placa de función dieléctrica ε_a y longitud d que está inmersa en un medio de función dieléctrica ε_b.

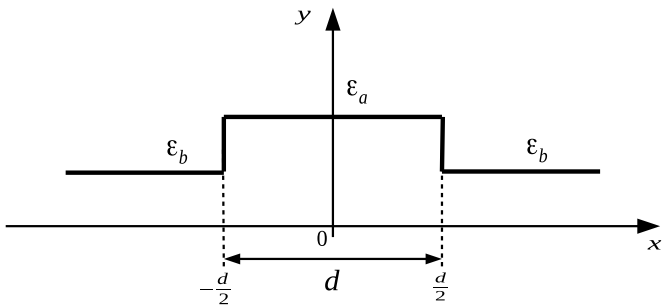

Figura 2.5: Función dieléctrica de una guía plana

2.3. Modos Transversal Magnético $[B_z(x, y, t) = 0]$

Se define una polarización Transversal Magnética (TM) en donde el campo magnético en la dirección z es cero de la forma $Bz(x, y, t) = 0$. Para obtener modos con la polarización (TM) guiados por la placa de material ε_a se escoge el campo eléctrico en cada uno de los medios de la forma

$$\mathbf{E}(\mathbf{x}, t) = \hat{k} E_z(x) \cos(k_y y) \cos(\omega t) \tag{2.80}$$

donde el vector de onda k_y y la frecuencia ω se conservan constantes en cada uno de los medios. Los términos $\cos(k_y y)$ y $\cos(\omega t)$ definen una propagación armónica en el espacio y en el tiempo, respectivamente. La dependencia del campo eléctrico en el eje x se escoge de la forma

$$E_z(x)=\begin{cases} Ae^{+k_{bx}\left(\frac{x+\frac{d}{2}}{d}\right)} & x<-\frac{d}{2} \\ B_1\cos(k_{ax}x)+B_2\sin(k_{ax}x) & -\frac{d}{2}<x<\frac{d}{2} \\ Ce^{-k_{bx}\left(\frac{x-\frac{d}{2}}{d}\right)} & x>\frac{d}{2} \end{cases} \tag{2.81}$$

Esta relación define que el campo al interior de la placa es armónico mientras que fuera de la placa el campo decae como una onda evanescente. Los vectores de onda k_{ax} y k_{bx} se obtienen a partir de las relaciones

$$k_{ax}=\sqrt{k_a^2-k_y^2}, \tag{2.82}$$

$$k_{bx}=\sqrt{k_y^2-k_b^2}. \tag{2.83}$$

Los vectores de onda en cada medio son

$$k_a=n_a\frac{\omega}{c}, \tag{2.84}$$

$$k_b=n_b\frac{\omega}{c}. \tag{2.85}$$

donde los índices de refracción son $n_a=\sqrt{\varepsilon_a\mu_a}$ y $n_b=\sqrt{\varepsilon_b\mu_b}$. El campo magnético asociado al campo eléctrico se obtiene a partir de la ecuación de Maxwell

$$\nabla\times\mathbf{E}(\mathbf{x},t)=-\frac{\mu}{c}\frac{\partial}{\partial t}\mathbf{H}(\mathbf{x},t) \tag{2.86}$$

Desarrollando el lado izquierdo de la ecuación 2.86 se tiene que el rotacional del campo eléctrico es

$$\nabla\times\mathbf{E}(\mathbf{x},t)=\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ 0 & 0 & E_z(x)cos(k_y y)cos(\omega t) \end{vmatrix}, \tag{2.87}$$

$$\nabla\times\mathbf{E}(\mathbf{x},t)=\left[E_z(x)cos(\omega t)\frac{\partial}{\partial y}cos(k_y y),-cos(k_y y)cos(\omega t)\frac{\partial}{\partial x}E_z(x),0\right], \tag{2.88}$$

$$\nabla\times\mathbf{E}(\mathbf{x},t)=-\left[k_y sen(k_y y)E_z(x),cos(k_y y)\frac{\partial}{\partial x}E_z(x),0\right]\cos(\omega t). \tag{2.89}$$

Y se escoge la forma temporal del campo magnético de la forma

$$\mathbf{H}(\mathbf{x},t)=\mathbf{H}(\mathbf{x})sen(\omega t) \tag{2.90}$$

De esta manera la derivada es

$$\frac{\partial}{\partial t}\mathbf{H}(\mathbf{x},t)=\omega cos(\omega t)\mathbf{H}(\mathbf{x}) \tag{2.91}$$

Entonces la ecuación 2.86 se puede escribir en la forma

$$\left[k_y sen(k_y y)E_z(x), cos(k_y y)\frac{\partial}{\partial x}E_z(x), 0\right]=\frac{\mu\omega}{c}\left[H_x(x,y),H_y(x,y),0\right] \tag{2.92}$$

Esta ecuación vectorial define un par de ecuaciones escalares

$$H_x(x,y)=\left[H_x(x)\right]\left[H_x(y)\right]=\left[E_z(x)\right]\left[\frac{c}{\mu\omega}k_y sen(k_y y)\right], \tag{2.93}$$

$$H_y(x,y)=\left[H_y(x)\right]\left[H_y(y)\right]=\left[\frac{\partial}{\partial x}E_z(x)\right]\left[\frac{c}{\mu\omega}cos(k_y y)\right]. \tag{2.94}$$

De esta manera se tienen las siguientes relaciones

$$\left[H_x(x)\right]=\left[E_z(x)\right], \tag{2.95}$$

$$\left[H_x(y)\right]=\left[\frac{c}{\mu\omega}k_y sen(k_y y)\right], \tag{2.96}$$

$$\left[H_y(x)\right]=\left[\frac{\partial}{\partial x}E_z(x)\right], \tag{2.97}$$

$$\left[H_y(y)\right]=\left[\frac{c}{\mu\omega}cos(k_y y)\right]. \tag{2.98}$$

Para la dependencia del campo magnético en el eje x se tiene $H_y(x)$ de la forma

$$H_y(x)=\begin{cases} Ak_{bx}e^{k_{bx}\left(\frac{x+\frac{d}{2}}{d}\right)} & x<-\frac{d}{2} \\ -B_1k_{ax}\sin(k_{ax}x)+B_2k_{ax}\cos(k_{ax}x) & -\frac{d}{2}<x<\frac{d}{2} \\ -Ck_{bx}e^{-k_{bx}\left(\frac{x-\frac{d}{2}}{d}\right)} & x>\frac{d}{2} \end{cases} \tag{2.99}$$

Retomando las ecuaciones 2.82 y 2.83 se tiene que para la existencia de ondas guiadas se requiere que la parte imaginaria de los vectores de onda k_{ax} y k_{bx} sean cero, es decir se requiere que

$$\mathrm{Im}(k_{ax})=0, \tag{2.100}$$

$$\mathrm{Im}(k_{bx}) = 0. \tag{2.101}$$

Estas condiciones establecen que la condición de modos es

$$k_a \geq k_y \geq k_b, \tag{2.102}$$

Es conveniente visualizar esta desigualdad para entender bien qué es lo que implica esta condición de modos y para hacer esto se requiere un poco de álgebra. Se comienza introduciendo el vector de onda reducido en cada uno de los materiales en la forma

$$Q_a = k_a \frac{d}{2\pi}, \tag{2.103}$$

$$Q_b = k_b \frac{d}{2\pi}. \tag{2.104}$$

La frecuencia reducida se define como

$$\Omega = \omega \frac{d}{2\pi c} \tag{2.105}$$

Se define que los vectores de onda en cada uno de los medios son

$$Q_a = n_a \Omega, \tag{2.106}$$

$$Q_b = n_b \Omega. \tag{2.107}$$

De esa manera se tiene que la condición de modos guiados de la ecuación 2.102 se define en términos de vectores de onda reducidos en la forma

$$Q_a \geq Q_y \geq Q_b \tag{2.108}$$

donde

$$Q_y = k_y \frac{d}{2\pi}. \tag{2.109}$$

En la Figura 2.6 se muestra una gráfica en donde se visualiza la región en donde pueden existir modos guiados. En el Apéndice A.3 se muestra el procedimiento que se ha realizado para hacer esta gráfica. La región con lineas rojas define la región en donde los campos oscilan en el medio ε_a. La región con lineas azules define la región en donde los modos decaen en el medio ε_b. La región en donde existen lineas rojas y azules define la combinación de (Q_y, Ω) en donde los modos guiados pueden existir.

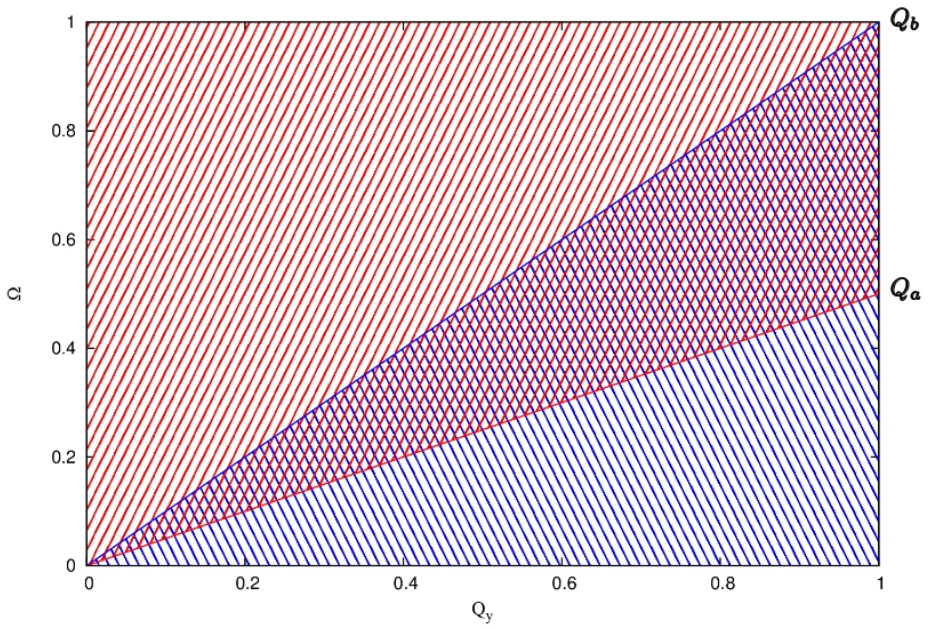

Figura 2.6: Región de modos guiados

2.3.1. Caso Par $(B_2 = 0)$

En la ecuación 2.81 cuando $B_2 = 0$ tenemos el caso en que el campo eléctrico tiene una polarización par con respecto al eje x como se muestra en la relación

$$E_z(x) = \begin{cases} Ae^{2\pi Q_{bx}\left(\frac{x+\frac{d}{2}}{d}\right)} & x < -\frac{d}{2} \\ B_1 \cos(2\pi Q_{ax} x) & -\frac{d}{2} < x < \frac{d}{2} \\ Ce^{-2\pi Q_{bx}\left(\frac{x-\frac{d}{2}}{d}\right)} & x > \frac{d}{2} \end{cases} \tag{2.110}$$

El campo magnético tangencial con respecto al eje x tiene la forma

$$H_y(x) = \begin{cases} A(\frac{2\pi}{d})Q_{bx}e^{2\pi Q_{bx}\left(\frac{x+\frac{d}{2}}{d}\right)} & x < -\frac{d}{2} \\ -B_1(\frac{2\pi}{d})Q_{ax} \sin(2\pi Q_{ax} x) & -\frac{d}{2} < x < \frac{d}{2} \\ -C(\frac{2\pi}{d})Q_{bx}e^{-2\pi Q_{bx}\left(\frac{x-\frac{d}{2}}{d}\right)} & x > \frac{d}{2} \end{cases} \tag{2.111}$$

La condición de continuidad del campo eléctrico tangencial en $x = -d/2$ se tiene

$$A = B_1 cos(\pi Q_{ax}) \tag{2.112}$$

La continuidad del campo magnético tangencial en $x = -d/2$ define la relación

$$AQ_{bx} = -B_1 Q_{ax} sin(-\pi Q_{ax}) \tag{2.113}$$

Dividiendo la ecuación 2.113 sobre 2.112 se tiene que la condición de modos pares es

$$\tan(\pi Q_{ax}) = \frac{Q_{bx}}{Q_{ax}} \tag{2.114}$$

Esta es una ecuación trascendente que es necesario resolver en forma numérica. Es conveniente escribirla en la forma

$$\tan(\pi\sqrt{n_a^2\Omega^2 - Q_y^2}) = \frac{\sqrt{Q_y^2 - n_b^2\Omega^2}}{\sqrt{n_a^2\Omega^2 - Q_y^2}} \tag{2.115}$$

En la Figura 2.7 se muestra cómo es posible encontrar los modos pares para el caso $\Omega = 1$.

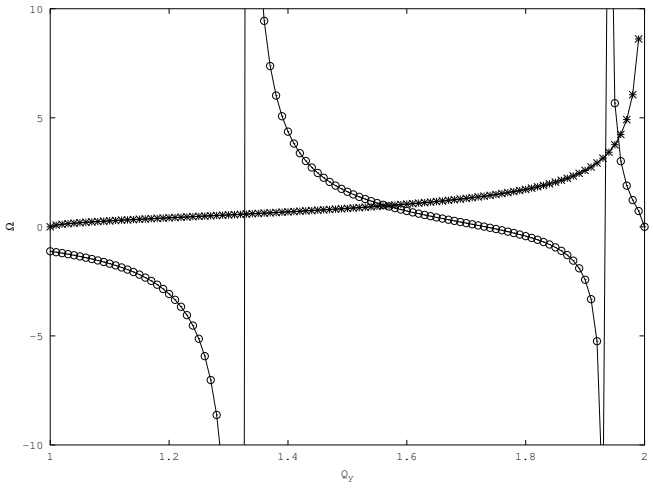

Figura 2.7: Modos pares para $\Omega = 1.0$

El programa con el que se ha realizado esta gráfica se encuentra en el Apéndice A.4. Si se ejecuta el programa para muchas frecuencias, se puede tener un mapa de los modos pares en el plano (Q_y, Ω).

2.3.2. Caso Impar $(B_1 = 0)$

En la ecuación 2.81 cuando $B_1 = 0$ se tiene el caso en que el campo eléctrico tiene una polarización impar con respecto al eje x como se muestra en la relación

$$E_z(x)=\begin{cases} Ae^{2\pi Q_{bx}\left(\frac{x+\frac{d}{2}}{d}\right)} & x<-\frac{d}{2} \\ B_2\sin(2\pi Q_{ax}x) & -\frac{d}{2}<x<\frac{d}{2} \\ Ce^{-2\pi Q_{bx}\left(\frac{x-\frac{d}{2}}{d}\right)} & x>\frac{d}{2} \end{cases} \tag{2.116}$$

El campo magnético tangencial con respecto al eje x tiene la forma

$$H_y(x)=\begin{cases} A(\frac{2\pi}{d})Q_{bx}e^{2\pi Q_{bx}\left(\frac{x+\frac{d}{2}}{d}\right)} & x<-\frac{d}{2} \\ B_1(\frac{2\pi}{d})Q_{ax}\cos(2\pi Q_{ax}x) & -\frac{d}{2}<x<\frac{d}{2} \\ -C(\frac{2\pi}{d})Q_{bx}e^{-2\pi Q_{bx}\left(\frac{x-\frac{d}{2}}{d}\right)} & x>\frac{d}{2} \end{cases} \tag{2.117}$$

La condición de continuidad del campo eléctrico tangencial en $x=-d/2$ se tiene

$$A=B_2\sin(-\pi Q_{ax}) \tag{2.118}$$

La continuidad del campo magnético tangencial en $x=-d/2$ define la relación

$$AQ_{bx}=B_1Q_{ax}\cos(-\pi Q_{ax}) \tag{2.119}$$

Dividiendo la ecuación 2.118 sobre 2.119 se tiene que la condición de modos impares es

$$\cot(\pi Q_{ax})=-\frac{Q_{bx}}{Q_{ax}} \tag{2.120}$$

Esta es una ecuación trascendente que es necesario resolver en forma numérica. En la Figura 2.8 se muestra cómo es posible encontrar los modos impares para el caso $\Omega=1$. El programa con el que se ha realizado esta gráfica se encuentra en el Apéndice A.4. Para esta frecuencia se encuentran dos soluciones. El primero corresponde al primer modo impar y el segundo al segundo modo impar.

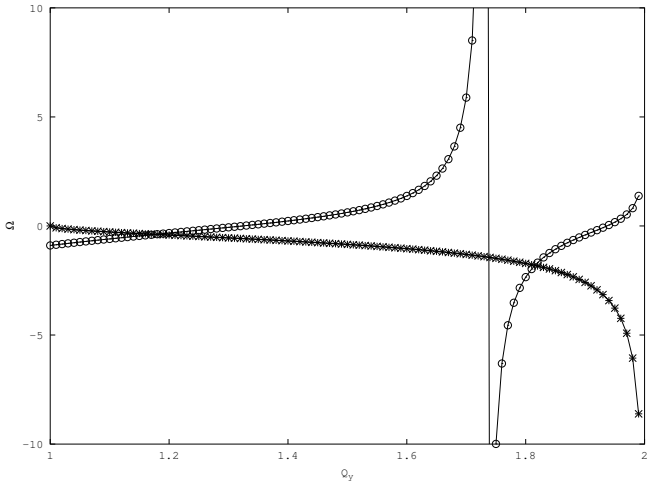

Figura 2.8: Modos impares para $\Omega = 1.0$

2.3.3. Relación de Dispersión

La siguiente figura muestra los modos pares e impares que se grafican utilizando las ecuaciones trascendentes descritas anteriormente. A diferencia de las otras figuras aquí se muestra un barrido en Ω y Q_y dentro de la región de las líneas de luz.

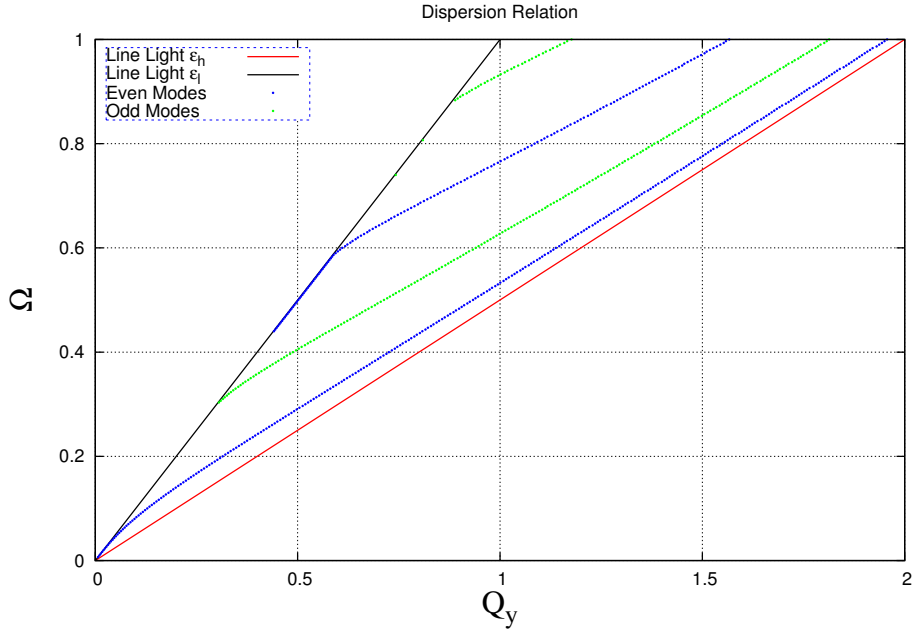

Figura 2.9: Relación de dispersión para una guía plana

2.4. Conclusiones

En conclusión, los resultados de este capítulo presentan la forma analítica de resolver determinados sistemas, se da a conocer el álgebra necesaria para describir los fenómenos que se quieren estudiar tomando como herramientas la física y matemáticas que se relacionan con cada paso del desarrollo.

Se establece la base teórica desde la cual se construyen los análisis numéricos y que muchas veces sirve para comprobar la eficacia de los mismos. En otras palabras este capítulo es la plataforma que sirve para analizar y estudiar los distintos sistemas propuestos.

2.5. Referencias

[1] Paloma Varela Nieto. *Orígenes del electromagnetismo. Oersted y Ampére.* Nivola Ediciones, 1900.

[2] José Antonio Díaz-Hellín Martínez del Rey. *El gran cambio en la Física. Faraday.* Nivola, 2001.

[3] W. Edward Gettys and Frederick Keller. *Fisica Clasica y Moderna (Spanish Edition).* McGraw-Hill Interamericana, 1999.

[4] Clerk Maxwell James; Sánchez Ron José Manuel ed. *Materia y Movimiento.* Crítica, 2006.

[5] Javier Ordoñez, Victor Navarro, and Jose Manuel Sanchez Ron. *Historia de la ciencia.* Espasa, 2013.

[6] Agustin Udias Vallina. *Historia de la Física. de Arquimides a Einstein.* Editorial Síntesis, 2010.

[7] John David Jackson. *Classical Electrodynamics, 2nd Edition.* Wiley, 1975.

[8] Robert Ehrlich. *Why Toast Lands Jelly-Side Down: Zen and the Art of Physics Demonstrations.* Princeton University Press, 1997.

3 El Método FDTD

El Método de Diferencias Finitas en el Dominio del Tiempo (FDTD) fue propuesto originalmente por Kane S. Yee en su artículo publicado en 1966 [1]. Yee propuso una solución discreta de las ecuaciones de Maxwell en base en aproximaciones de diferencias centrales de las derivadas espaciales y temporales de las ecuaciones rotacionales. La novedad del enfoque de Yee fue el escalonamiento de los campos eléctricos y magnéticos en el espacio y tiempo. Yee derivó una formulación tridimensional completa, y validó el método con problemas bidimensionales. El método de Yee pasó mayormente inadvertido por casi una década. Por último, en 1975, Taflove y Brodwin aplican el método de Yee para simular la dispersión por cilindros dieléctricos [2] y calentamiento biológico [3], y en 1977, Holland aplica el método para predecir las corrientes inducidas en una aeronave por un pulso electromagnético [4]. Holland resumió de esta manera la razón de la demora de la aplicación del método de Yee:

El algoritmo fue descrito por primera vez por Yee hace unos 10 años. En ese momento, no existían las computadoras que podrían implementar este esquema prácticamente en tres dimensiones. Esto ya no es el caso. Ahora es posible modelar un espacio de interés en una malla de 30x30x30 y tener acceso aleatorio a las 162,000 cantidades de campo resultantes. Estos avances en tecnología computacional han llevado a Longmire a acreditar el algoritmo de Yee por ser "La forma numérica más rápida de resolver las ecuaciones de Maxwell [5]."

Las ecuaciones diferenciales parciales de Maxwell, formuladas hace más de 150 años, representan la unificación fundamental de los campos eléctricos y magnéticos que predicen fenómenos de ondas electromagnéticas que el premio nobel Richard Feynman ha llamado el logro más sobresaliente de la ciencia del siglo XIX. En la actualidad, los ingenieros y científicos de todo el mundo utilizan computadoras que van desde máquinas de escritorio hasta arreglos masivos de procesadores en paralelo para obtener soluciones a estas ecuaciones con el propósito de investigar guías de onda electromagnéticas, radiación y fenómenos de dispersión [6].

3.1. Las Ecuaciones de Maxwell en un Medio Dieléctrico

Se plantea la propagación de las ondas electromagnéticas a través de las ecuaciones rotacionales de Maxwell y las relaciones constitutivas en la forma

$$\frac{\partial}{\partial t}\mathbf{B}(\mathbf{x},t) = -c\nabla \times \mathbf{E}(\mathbf{x},t), \tag{3.1}$$

$$\mathbf{H}(\mathbf{x},t) = \frac{1}{\mu(\mathbf{x})}\mathbf{B}(\mathbf{x},t), \tag{3.2}$$

$$\frac{\partial}{\partial t}\mathbf{D}(\mathbf{x},t) = c\nabla \times \mathbf{H}(\mathbf{x},t), \tag{3.3}$$

$$\mathbf{E}(\mathbf{x},t) = \frac{1}{\varepsilon(\mathbf{x})}\mathbf{D}(\mathbf{x},t). \tag{3.4}$$

3.1.1. Tratamiento Analítico de las Ecuaciones Diferenciales

Se estudia una onda electromagnética propagándose en el vacío en una dimensión. Se considera una polarización de la forma

$$\mathbf{E}(\mathbf{x},t) = \begin{bmatrix} 0 \\ 0 \\ E_z(y,t) \end{bmatrix} \tag{3.5}$$

Se procede a desarrollar el rotacional del campo eléctrico

$$\nabla \times \mathbf{E}(\mathbf{x},t) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ 0 & 0 & E_z(y,t) \end{vmatrix} = \frac{\partial}{\partial y}E_z(y,t)\hat{j} \tag{3.6}$$

De esta manera la ecuación de Faraday se escribe de la forma

$$\frac{\partial}{\partial t}B_x(y,t) = -c\frac{\partial}{\partial y}E_z(y,t) \tag{3.7}$$

Por otra parte, para la ecuación de Ampere-Maxwell se desarrolla el rotacional

$$\nabla \times \mathbf{H}(\mathbf{x},t) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ H_x(y,t) & 0 & 0 \end{vmatrix} = -\frac{\partial}{\partial y}H_x(y,t)\hat{k} \tag{3.8}$$

De esta forma se tiene la relación

$$\frac{\partial}{\partial t}D_z(y,t) = -c\frac{\partial}{\partial y}H_x(y,t) \tag{3.9}$$

Es conveniente escribir las ecuaciones en una forma recursiva ya que esto nos permite definir un algoritmo computacional para resolverlas numéricamente, las ecuaciones se reescriben de la forma

$$\frac{\partial}{\partial t}D_z(y,t) = -c\frac{\partial}{\partial y}H_x(y,t), \tag{3.10}$$

$$E_z(y,t) = \frac{1}{\varepsilon(y)}D_z(y,t), \tag{3.11}$$

$$\frac{\partial}{\partial t}B_x(y,t) = -c\frac{\partial}{\partial y}E_z(y,t), \tag{3.12}$$

$$H_x(y,t) = \frac{1}{\mu(y)}B_x(y,t). \tag{3.13}$$

Las ecuaciones (3.10-3.13) definen un sistema de ecuaciones que puede ser resuelto mediante un algoritmo computacional como se muestra en la figura 3.1. Para lograr una formulación discreta de estas ecuaciones, es necesario aproximar las derivadas espacial y temporal en forma de diferencias finitas centrales [7].

$$\frac{d}{dy}f(y) \simeq \frac{f(y+\Delta y/2) - f(y-\Delta y/2)}{\Delta y} \tag{3.14}$$

La idea general del método FDTD es la de expresar las ecuaciones diferenciales en términos de diferencias finitas. De esta forma, es conveniente considerar que los valores continuos del espacio y del tiempo sean ahora tomados en forma discreta por medio de las relaciones $y = j\Delta y$ y $t = n\Delta t$, respectivamente. Se considera la derivada temporal de la forma

$$\frac{\partial}{\partial t}D_z(y,t) \simeq \frac{D_z(y,t+\Delta t/2) - D_z(y,t-\Delta t/2)}{\Delta t} \tag{3.15}$$

De forma similar se tiene

$$\frac{\partial}{\partial y}H_x(y,t) \simeq \frac{H_x(y+\Delta y/2,t) - H_x(y-\Delta y/2,t)}{\Delta y} \tag{3.16}$$

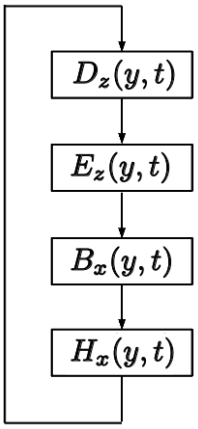

Figura 3.1: Algoritmo de actualización de campos

Para simplificar la notación, se agrega como superíndice la dependencia temporal del campo en la forma $D_z[j\Delta y,(n+1/2)\Delta t] \rightarrow D_z^{n+1/2}(j)$. La ecuación 3.10 se escribe en forma discreta

$$\frac{D_z^{n+1/2}(j) - D_z^{n-1/2}(j)}{\Delta t} = -c\frac{H_x^n(j+1/2) - H_x^n(j-1/2)}{\Delta y} \tag{3.17}$$

Para poner la forma discreta de la ecuación 3.12 es necesario considerar el punto $(y,t) = (y+\Delta y/2, t+\Delta t/2)$ ya que solo así se puede obtener un sistema de ecuaciones intercalado en el espacio y en el tiempo.

$$\frac{B_x^{n+1}(j+1/2) - B_x^n(j+1/2)}{\Delta t} = -c\frac{E_z^{n+1/2}(j+1) - E_z^{n+1/2}(j)}{\Delta y} \tag{3.18}$$

Se consideran las ecuaciones para crear un sistema de ecuaciones acoplado de la forma

$$D_z^{n+1/2}(j) = D_z^{n-1/2}(j) - c\frac{\Delta t}{\Delta y}\left[H_x^n(j+1/2) - H_x^n(j-1/2)\right], \tag{3.19}$$

$$B_x^{n+1}(j+1/2) = B_x^n(j+1/2) - c\frac{\Delta t}{\Delta z}\left[E_z^{n+1/2}(j+1) - E_z^{n+1/2}(j)\right]. \tag{3.20}$$

De acuerdo a la condición de Courant–Friedrich–Levy. La que se define como una condición de convergencia de ecuaciones diferenciales en derivadas parciales que establece que el paso de tiempo debe ser inferior a un cierto valor para mantener la estabilidad de la simulación [6]. Se tiene que $c\Delta t/\Delta y = 1/2$, de esta forma

$$D_z^{n+1/2}(j) = D_z^{n-1/2}(j) - \frac{1}{2}\left[H_x^n(j+1/2) - H_x^n(j-1/2)\right], \tag{3.21}$$

$$E_z^{n+1/2}(j) = \frac{1}{\varepsilon(j)}D_z^{n+1/2}(j), \tag{3.22}$$

$$B_x^{n+1}(j+1/2) = B_x^n(j+1/2) - \frac{1}{2}\left[E_z^{n+1/2}(j+1) - E_z^n(j)\right], \tag{3.23}$$

$$H_x^{n+1}(j+1/2) = \frac{1}{\mu(j+1/2)}B_x^{n+1/2}(j+1/2). \tag{3.24}$$

3.1.2. Programación de las Ecuaciones Discretas

Se programan las ecuaciones discretas en una dimensión definiendo una fuente emitiendo campo en el espacio libre como se muestra a continuación

Listado 3.1: Ecuaciones discretas en una dimensión

```
1 #include <stdio.h>
2 #include <math.h>
3 #include <iostream>
4 #include <fstream>
```

```
5  #include <sstream>
6
7  using namespace std;
8  int main() {
9      int Ny = 100, Nt = 50;
10     int iy, it;
11     float pi, c;
12     float l0, tao0, w0;
13     float dy, dt, t;
14     float yV[Ny], Ez[Ny], Dz[Ny], Hx[Ny], Bx[Ny];
15     float epsi[Ny], mu[Ny];
16
17     pi = 2.0 * asin(1.0);
18     c = 3.0e8;
19     l0 = 630e-9;
20     tao0 = l0 / c;
21     w0 = (2.0 * pi) / tao0;
22     dy = l0 / 10.0;
23     dt = dy / (2.0 * c);
24
25     for (iy = 0; iy <= Ny; iy++) {
26         yV[iy] = iy * dy;
27         Ez[iy] = 0.0;
28         Hx[iy] = 0.0;
29         Dz[iy] = 0.0;
30         Bx[iy] = 0.0;
31         epsi[iy] = 1.0;
32         mu[iy] = 1.0;
33     }
34
35     for (it = 0; it <= Nt; it++) {
36         t = it * dt;
37         for (iy = 1; iy <= Ny; iy++) {
38             Dz[iy] = Dz[iy] - 0.5 * (Hx[iy] - Hx[iy - 1]);
39             Ez[iy] = (1.0 / epsi[iy]) * Dz[iy];
40         }
41
42         Ez[Ny / 2] = Ez[Ny / 2] + sin(w0 * t);
43         for (iy = 0; iy <= (Ny - 1); iy++) {
44             Bx[iy] = Bx[iy] - 0.5 * (Ez[iy + 1] - Ez[iy]);
45             Hx[iy] = (1.0 / mu[iy]) * Bx[iy];
46         }
47     }
48
49     ofstream myfile;
50     myfile.open("data.dat");
51     for (iy = 0; iy <= Ny; iy++) {
52         myfile << iy << " " << yV[iy] / l0 << " " << Ez[iy] << " " << Hx[iy] << "\n";
53     }
54     myfile.close();
55     return 0;
56 }
```

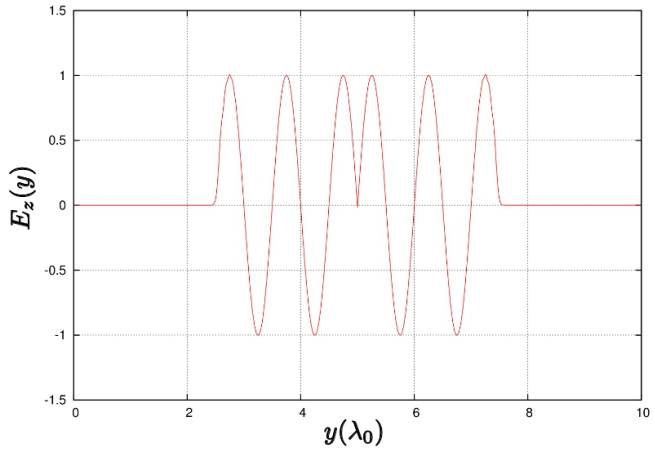

Figura 3.2: Campo eléctrico debido a una fuente de la forma $E(t) = \Theta(t)sin(\omega_0 t)$

3.2. Propagación del Campo Electromagnético en una Multicapa

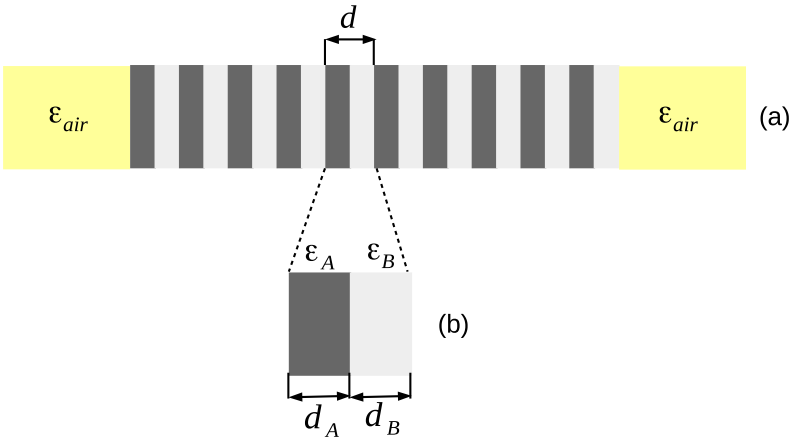

Figura 3.3: (a) Multicapa finita compuesta por la repetición periódica de 10 celdas unitarias de espesor d**. (b) Celda unitaria compuesta por dos películas delgadas con alta (**ε_A**) y baja (**ε_B**) función dieléctrica y espesores** d_A **y** d_B**, respectivamente.**

En esta sección se analiza la propagación del campo electromagnético a través de una multicapa. En la Figura 3.3(a) se muestra una multicapa finita compuesta por una repetición periódica de 10 celdas unitarias de espesor d rodeadas en ambos lados por aire (ε_{air}). La celda unitaria se define en la Figura 3.3(b), la cual contiene dos películas delgadas con función dieléctrica alta (ε_A) y función dieléctrica baja (ε_B). Estas películas delgadas tienen un espesor d_A y d_B, respectivamente.

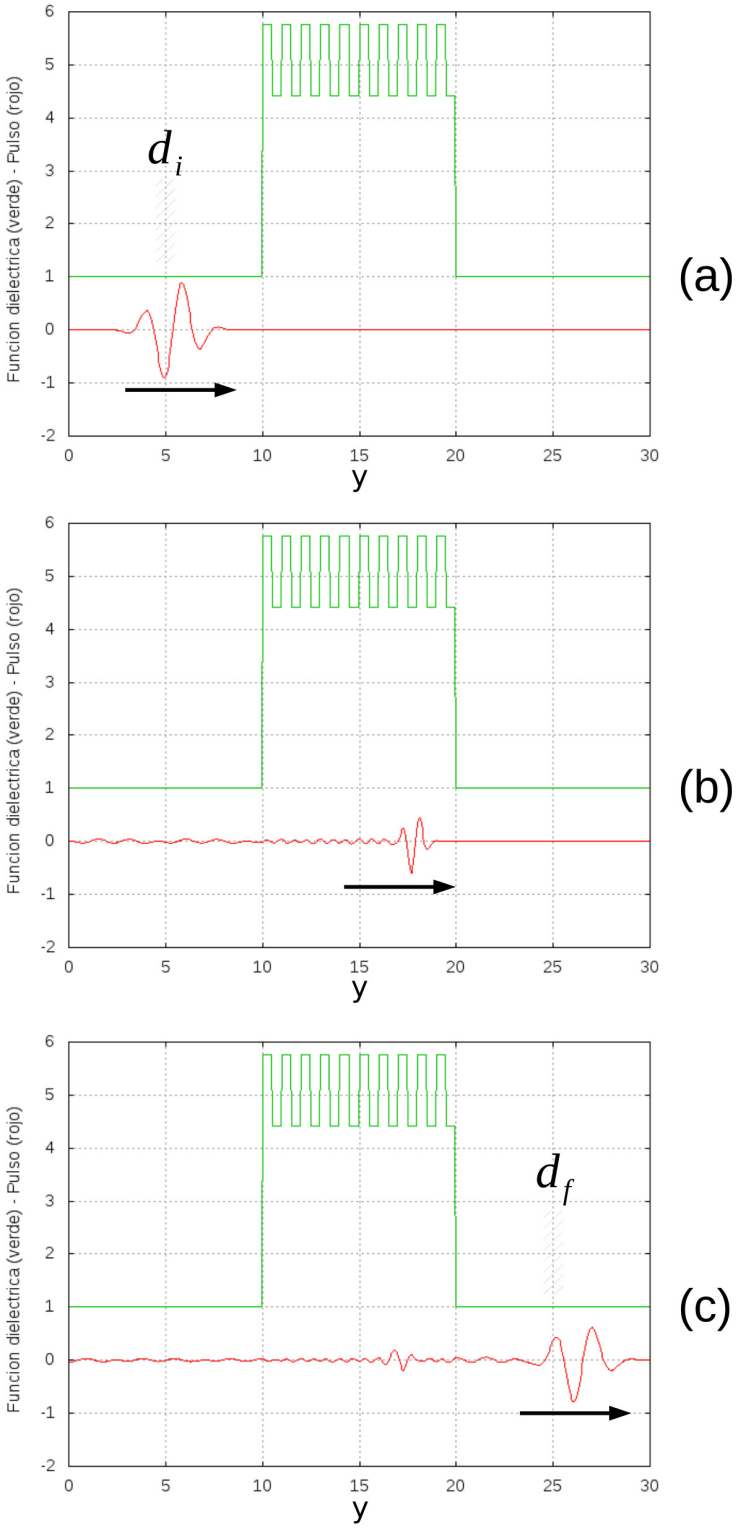

Figura 3.4: Propagación de un pulso a través de una multicapa. (a) Pulso incidiendo de izquierda a derecha hacia una mulicapa. (b) Pulso que viaja al interior de una multicapa. (c) Pulso que viaja hacia afuera de la multicapa.

Se considera la propagación a través de una multicapa de funciones dieléctricas $\varepsilon_A = 5.76$ y $\varepsilon_B = 4.41$, que corresponden a los parámetros materiales de silicio poroso de alta y baja densidad, según hemos reportado en la Ref. [8]. Los espesores son $d_A = d_B = d/2$. Se considera la propagación de un pulso que viaja de izquierda a derecha, según lo ilustra la Figura 3.4. Se muestra en verde la función dieléctrica como función de la posición, $\varepsilon(x)$. De la misma forma, se muestra en rojo el perfil del campo eléctrico $E_z(x,t)$. En el panel (a) se muestra cómo un pulso electromagnético se aproxima a una multicapa. Este pulso pasa a través del detector d_i. En el panel (b) se ilustra la propagación del pulso al interior de la multicapa, la cual se caracteriza por una propagación del campo electromagnético que viaja de izquierda a derecha. Finalmente, en el panel (c) se presenta el pulso que sale de la multicapa. Este pulso pasa a través del detector d_f.

El pulso incidente está compuesto por una componente gaussiana y una componente sinusoidal de la forma

$$E_i(t) = E_o \sin(\omega t) \exp\left[-\frac{1}{2}\frac{(t-t_0)^2}{\sigma^2}\right] \tag{3.25}$$

donde ω es la componente monocromática del pulso, t_0 define el centro temporal del pulso gaussiano, σ es el ancho del pulso temporal. La transformada de Fourier se define mediante la relación

$$E_z(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} E_z(t) e^{i\omega t} dt \tag{3.26}$$

Es conveniente definir el periodo $\tau_d = c/d$ para la normalización de los pulsos temporales en las siguientes gráficas.

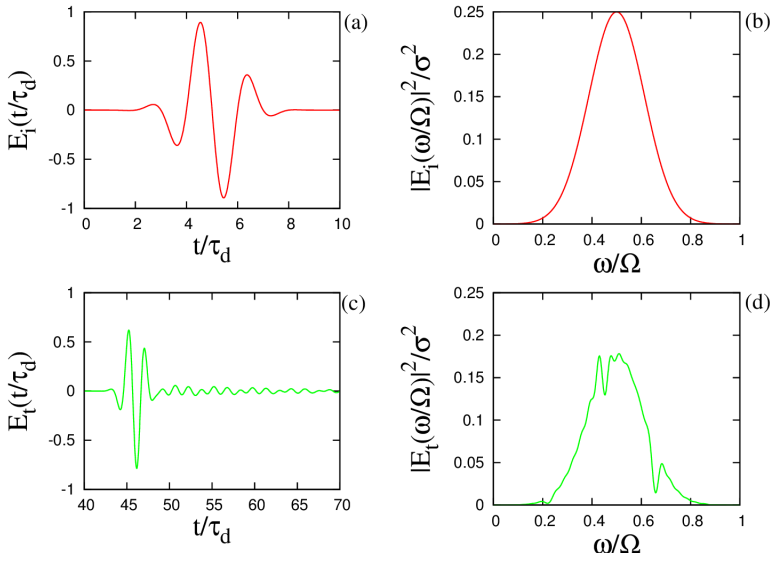

Figura 3.5: Pulsos electromagnéticos. (a) Distribución temporal del pulso incidente E_i(t). (b) Distribución en frecuencias del pulso incidente $E_i(\omega)$. (c) Distribución temporal del pulso transmitido E_t(t). (d) distribución en frecuencias del pulso transmitido $E_t(\omega)$

En la Figura 3.5 se muestra la distribución temporal y en frecuencias de los pulsos incidente y transmitido. En el panel (a) se muestra la distribución temporal del pulso como es medido por el detector d_i, que se ilustra en la Figura 3.4(a). La frecuencia del pulso es $\omega = \Omega/2$ y el ancho del pulso es $\sigma = \tau_d$. En el panel (b) se muestra la transformada de Fourier del pulso incidente. Se observa que se tiene una gaussiana centrada en $\omega = \Omega/2$ y ancho σ. En el panel (c) se muestra la distribución temporal del pulso transmitido y que es medido por el detector d_f, según lo ilustra la Figura 3.4(c). Se observa que solo hay campo a partir de $t \simeq 43\tau_d$. La amplitud del pulso está concentrada en el rango $43\tau_d < t < 48\tau_d$. Para tiempos $t > 48\tau_d$, la amplitud del campo eléctrico es muy pequeña. En el panel (d) se muestra la Transformada de Fourier del pulso transmitido. Se observa que tiene una forma similar a un pulso gaussiano, pero existen dos claros valles en las frecuencias $\omega = 0.22\Omega$ y $\omega = 0.68\Omega$.

La transmisión a través de la multicapa se calcula a través de la relación

$$T(\omega) = \frac{|E_t(\omega)|^2}{|E_i(\omega)|^2} \tag{3.27}$$

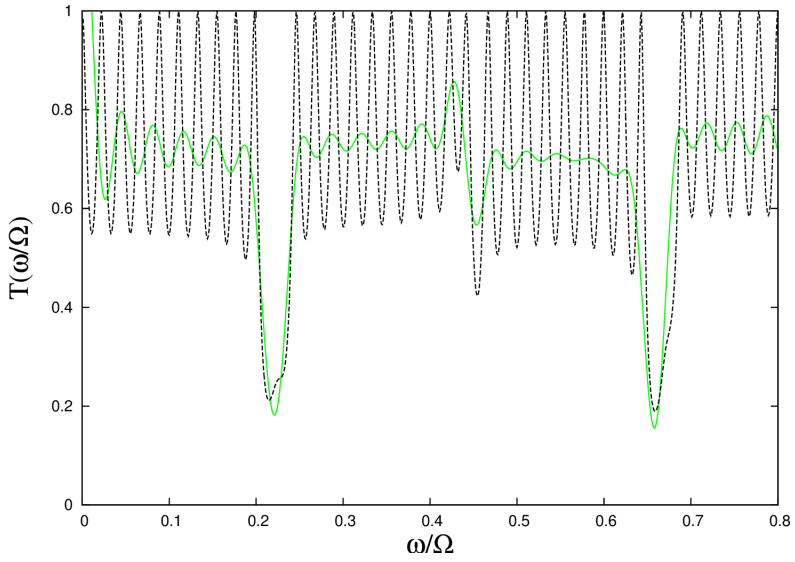

Figura 3.6: Transmisión a través de una multicapa.

En la Figura 3.6 se muestra la transmisión a través de la multicapa. Con línea verde se presenta la transmisión $T(\omega)$ calculada con el método FDTD. Con línea negra a trozos se muestra la transmisión calculada con un método exacto. Se observa que el método FDTD es capaz de definir adecuadamente la presencia de gaps en la región de bandas prohibidas y además, la presencia de oscilaciones en la regiones de banda permitida.

En la Figura 3.7 se muestra un mapa de bandas que fue publicado en la Figura 1 de la Ref. [8]. El factor de llenado se define como la región ocupada por el medio de alta función dieléctrica en la celda unitaria $f = d_A/d$. La región en gris define las bandas prohibidas para campos electromagnéticos

en una multicapa. Es posible observar la relación de la Figura 3.6 con la Figura 3.7, considerando un factor de llenado $f = 0.5$, donde coincide la región de bandas prohibidas y bandas permitidas.

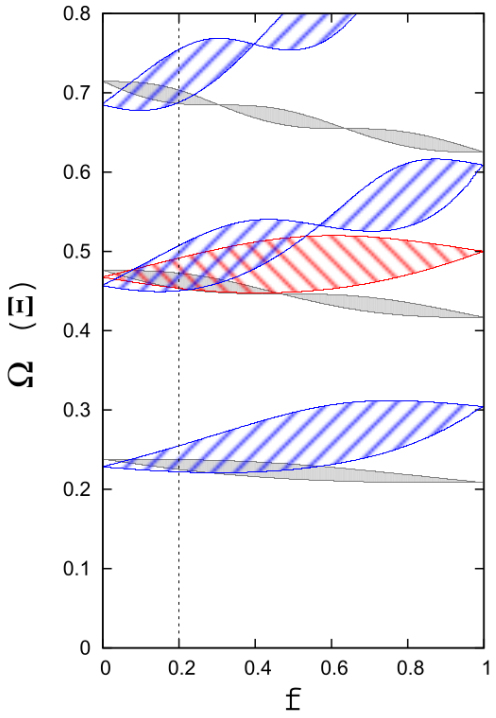

Figura 3.7: Mapa de bandas como función del factor de llenado f. Las regiones en gris definen la región de bandas prohibidas para los campos electromagnéticos.

3.3. Propagación de Ondas Electromagnéticas en Dos Dimensiones Para el Caso TM

El caso Transversal Magnético (TM) se define por la existencia de los campos $E_z(x, y, t)$, $H_x(x, y, t)$ y $H_y(x, y, t)$ [9].

3.3.1. Tratamiento Analítico de las Ecuaciones Diferenciales

Se estudia una onda electromagnética propagándose en el vacío en dos dimensiones. Se inicia el análisis definiendo la polarización para el campo eléctrico de la forma

$$\mathbf{E}(\mathbf{x}, t) = \hat{k} E_z(x, y, t) \tag{3.28}$$

El rotacional del campo eléctrico es

$$\nabla \times \mathbf{E}(\mathbf{x},t) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ 0 & 0 & E_z(x,y,t) \end{vmatrix} \tag{3.29}$$

Los campos magnéticos asociados son obtenidos a través de las ecuaciones

$$\frac{\partial}{\partial t} B_x(x,y,t) = -c\frac{\partial}{\partial y} E_z(x,y,t), \tag{3.30}$$

$$\frac{\partial}{\partial t} B_y(x,y,t) = +c\frac{\partial}{\partial x} E_z(x,y,t). \tag{3.31}$$

La relación constitutiva magnética da lugar a las siguientes relaciones

$$H_x(x,y,t) = \frac{1}{\mu(x,y)} B_x(x,y,t), \tag{3.32}$$

$$H_y(x,y,t) = \frac{1}{\mu(x,y)} B_y(x,y,t). \tag{3.33}$$

El rotacional para el campo magnético es

$$\nabla \times \mathbf{H}(\mathbf{x},t) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ H_x(x,y,t) & H_y(x,y,t) & 0 \end{vmatrix} \tag{3.34}$$

La ecuación de Ampére-Maxwell da lugar a la relación

$$\frac{\partial}{\partial t} D_z(x,y,t) = c\frac{\partial}{\partial x} H_y(x,y,t) - c\frac{\partial}{\partial y} H_x(x,y,t) \tag{3.35}$$

La relación constitutiva eléctrica da lugar a la relación

$$E_z(x,y,t) = \frac{1}{\varepsilon(x,y)} D_z(x,y,t) \tag{3.36}$$

Es conveniente escribir las ecuaciones en una forma recursiva ya que esto nos permite definir un algoritmo computacional para resolverlas numéricamente, las ecuaciones se reescriben de la forma

$$\frac{\partial}{\partial t} D_z(x,y,t) = c\frac{\partial}{\partial x} H_y(x,y,t) - c\frac{\partial}{\partial y} H_x(x,y,t), \tag{3.37}$$

$$E_z(x,y,t) = \frac{1}{\varepsilon(x,y)} D_z(x,y,t), \tag{3.38}$$

$$\frac{\partial}{\partial t} B_x(x,y,t) = -c\frac{\partial}{\partial y} E_z(x,y,t), \tag{3.39}$$

$$H_x(x,y,t) = \frac{1}{\mu(x,y)} B_x(x,y,t), \tag{3.40}$$

$$\frac{\partial}{\partial t} B_y(x,y,t) = c \frac{\partial}{\partial x} E_z(x,y,t), \tag{3.41}$$

$$H_y(x,y,t) = \frac{1}{\mu(x,y)} B_y(x,y,t). \tag{3.42}$$

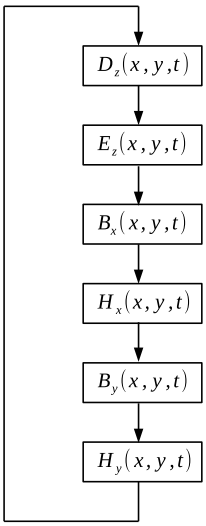

Figura 3.8: Algoritmo de actualización de campos

3.3.2. La Discretización de las Ecuaciones Diferenciales

Se discretiza la primer ecuación en el punto $(x,y,t) = (\Delta x, \Delta y, \Delta t)$

$$\frac{D_z^{n+1/2}(i,j) - D_z^{n-1/2}(i,j)}{\Delta t} = c\frac{H_y^n(i+1/2,j) - H_y^n(i-1/2,j)}{\Delta x} - c\frac{H_x^n(i,j+1/2) - H_x^n(i,j-1/2)}{\Delta y} \tag{3.43}$$

Una ecuación rotacional se discretiza en el punto $(x,y,t) = [i\Delta x, (j+1/2)\Delta y, (n+1/2)\Delta t]$

$$\frac{B_x^{n+1}(i,j+1/2) - B_x^n(i,j+1/2)}{\Delta t} = -c\frac{E_z^{n+1/2}(i,j+1) - E_z^{n+1/2}(i,j)}{\Delta y} \tag{3.44}$$

La otra ecuación rotacional se discretiza en el punto $(x,y,t) = [(i+1/2)\Delta x, j\Delta y, (n+1/2)\Delta t]$

$$\frac{B_y^{n+1}(i+1/2,j) - B_y^n(i+1/2,j)}{\Delta t} = +c\frac{E_z^{n+1/2}(i+1,j) - E_z^{n+1/2}(i,j)}{\Delta x} \tag{3.45}$$

La actualización de los campos se plantea como un algoritmo (ver Figura 3.8) de la siguiente manera

$$D_z^{n+1/2}(i,j) = D_z^{n-1/2}(i,j) + \frac{1}{2}[H_y^n(i+1/2,j) - H_y^n(i-1/2,j)] - \frac{1}{2}[H_x^n(i,j+1/2) - H_x^n(i,j-1/2)], \tag{3.46}$$

$$E_z^{n+1/2}(i,j) = \frac{1}{\varepsilon(i,j)} D_z^{n+1/2}(i,j), \tag{3.47}$$

$$B_x^{n+1}(i,j+1/2) = B_x^n(i,j+1/2) - \frac{1}{2}[E_z^{n+1/2}(i,j+1) - E_z^{n+1/2}(i,j)], \tag{3.48}$$

$$H_x^{n+1}(i,j+1/2) = \frac{1}{\mu(i,j+1/2)} B_x^{n+1}(i,j+1/2), \tag{3.49}$$

$$B_y^{n+1}(i+1/2,j) = B_y^n(i+1/2,j) + \frac{1}{2}[E_z^{n+1/2}(i+1,j) - E_z^{n+1/2}(i,j)], \tag{3.50}$$

$$H_y^{n+1}(i+1/2,j) = \frac{1}{\mu(i+1/2)} B_y^{n+1}(i+1/2,j). \tag{3.51}$$

3.3.3. La Programación de las Ecuaciones Discretas

Se programan las ecuaciones discretas en dos dimensiones definiendo una malla de simulación de tamaño $100x100$ constituida por aire. Se programa una fuente gaussiana localizada en el punto $x = 50$ y $y = 50$. La fuente genera un pulso que se propaga por 100 periodos de tiempo. El listado del programa se muestra a continuación

Listado 3.2: Ecuaciones discretas en dos dimensiones (tm)

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <stdlib.h>
4  #include <iostream>
5  #include <fstream>
6
7  using namespace std;
8
9  #define Nx 100
10 #define Ny 100
11 #define Nt 100
12
13 int main() {
14     int i, j, n;
15     float pi, c;
16     float l0, tao0, w0;
17     float dx, dy, dt;
18     float t, t0, sigma;
19     float Ez[Nx][Ny], Dz[Nx][Ny];
```

```
20      float Hx[Nx][Ny], Bx[Nx][Ny];
21      float Hy[Nx][Ny], By[Nx][Ny];
22      float epsi[Nx][Ny], mu_x[Nx][Ny], mu_y[Nx][Ny];
23
24      pi = 2.0 * asin(1.0);
25      c = 3.0e8;
26
27      l0 = 630.0e9;
28      tao0 = l0 / c;
29      w0 = (2.0 * pi) / tao0;
30
31      t0 = 2.0 * tao0;
32      sigma = 0.5 * tao0;
33
34      dx = l0 / 10.0;
35      dy = dx;
36      dt = dx / (2.0 * c);
37
38      for (i = 0; i < Nx; i++) {
39          for (j = 0; j < Ny; j++) {
40              Ez[i][j] = 0.0;
41              Dz[i][j] = 0.0;
42              Hx[i][j] = 0.0;
43              Bx[i][j] = 0.0;
44              Hy[i][j] = 0.0;
45              By[i][j] = 0.0;
46              epsi[i][j] = 1.0;
47              mu_x[i][j] = 1.0;
48              mu_y[i][j] = 1.0;
49          }
50      }
51
52      for (n = 0; n < Nt; n++) {
53          t = n * dt;
54          for (i = 0; i < Nx; i++) {
55              for (j = 0; j < Ny; j++) {
56                  Dz[i][j] = Dz[i][j] + 0.5 * (Hy[i][j] - Hy[i - 1][j])
57                          - 0.5 * (Hx[i][j] - Hx[i][j - 1]);
58                  Ez[i][j] = (1.0 / epsi[i][j]) * Dz[i][j];
59              }
60          }
61
62          Ez[Nx / 2][Ny / 2] = Ez[Nx / 2][Ny / 2]
63                  + exp(-0.5 * ((t - t0) / sigma) * ((t - t0) / sigma));
64          cout << n << " " << Ez[Nx / 2][Ny / 2] << " "
65                  << exp(-0.5 * ((t - t0) / sigma) * ((t - t0) / sigma)) << "\n";
66
67          for (i = 0; i < Nx; i++) {
68              for (j = 0; j < Ny; j++) {
69                  Bx[i][j] = Bx[i][j] - 0.5 * (Ez[i][j + 1] - Ez[i][j]);
70                  Hx[i][j] = (1.0 / mu_x[i][j]) * Bx[i][j];
71              }
72          }
73
74          for (i = 0; i < Nx; i++) {
75              for (j = 0; j < Ny; j++) {
76                  By[i][j] = By[i][j] + 0.5 * (Ez[i + 1][j] - Ez[i][j]);
77                  Hy[i][j] = (1.0 / mu_y[i][j]) * By[i][j];
78              }
```

```
79          }
80
81      } // ciclo temporal
82
83      ofstream myfile;
84      myfile.open("data.dat");
85      for (i = 0; i < Nx; i++) {
86          for (j = 0; j < Ny; j++) {
87              myfile << i << "  " << j << "    " << Ez[i][j] << "\n";
88          }
89      }
90      myfile.close();
91
92      return 0;
93      cout << pi << "\n";
94  }
```

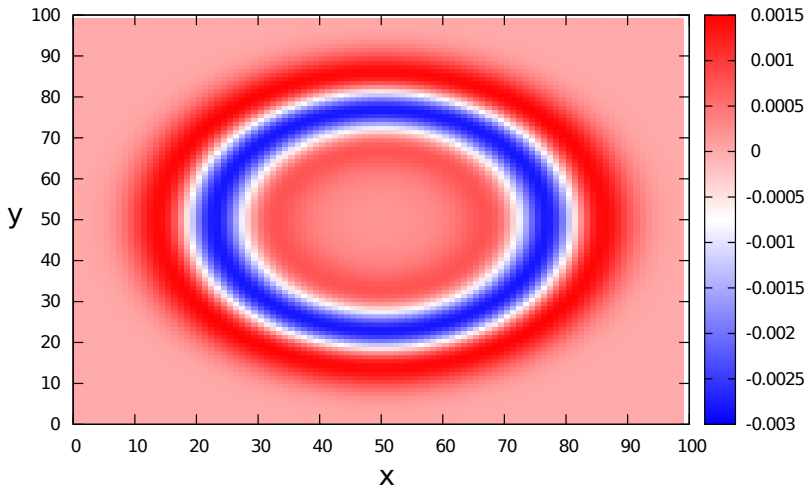

Figura 3.9: Campo eléctrico en dos dimensiones

3.4. Propagación de Ondas Electromagnéticas en Dos Dimensiones Para el Caso TE

El caso TE se define por la existencia de los campos $H_z(x, y, t)$, $E_x(x, y, t)$ y $E_y(x, y, t)$ [9].

3.4.1. Tratamiento Analítico de las Ecuaciones Diferenciales

Se inicia el análisis definiendo el campo magnético de la forma

$$\mathbf{H}(\mathbf{x}, t) = \hat{k} H_z(x, y, t) \tag{3.52}$$

El rotacional del campo es

$$\nabla \times \mathbf{H}(\mathbf{x},t) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ 0 & 0 & H_z(x,y,t) \end{vmatrix} \tag{3.53}$$

Los campos eléctricos asociados son obtenidos a través de las ecuaciones

$$\frac{\partial}{\partial t} D_x(x,y,t) = +c\frac{\partial}{\partial y} H_z(x,y,t), \tag{3.54}$$

$$\frac{\partial}{\partial t} D_y(x,y,t) = -c\frac{\partial}{\partial x} H_z(x,y,t). \tag{3.55}$$

La relación constitutiva da lugar a las siguientes relaciones

$$E_x(x,y,t) = \frac{1}{\varepsilon(x,y)} D_x(x,y,t), \tag{3.56}$$

$$E_y(x,y,t) = \frac{1}{\varepsilon(x,y)} D_y(x,y,t). \tag{3.57}$$

El rotacional para el campo eléctrico es

$$\nabla \times \mathbf{E}(\mathbf{x},t) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ E_x(x,y,t) & E_y(x,y,t) & 0 \end{vmatrix} \tag{3.58}$$

La ecuación de Faraday da lugar a la relación

$$\frac{\partial}{\partial t} B_z(x,y,t) = -c\frac{\partial}{\partial x} E_y(x,y,t) + c\frac{\partial}{\partial y} E_x(x,y,t) \tag{3.59}$$

La relación constitutiva magnética da lugar a la relación

$$H_z(x,y,t) = \frac{1}{\mu(x,y)} B_z(x,y,t) \tag{3.60}$$

Es conveniente escribir las ecuaciones en una forma recursiva ya que esto nos permite definir un algoritmo computacional para resolverlas numéricamente, las ecuaciones se reescriben de la forma

$$\frac{\partial}{\partial t} B_z(x,y,t) = -c\frac{\partial}{\partial x} E_y(x,y,t) + c\frac{\partial}{\partial y} E_x(x,y,t), \tag{3.61}$$

$$H_z(x,y,t) = \frac{1}{\mu(x,y)} B_z(x,y,t), \tag{3.62}$$

$$\frac{\partial}{\partial t} D_x(x,y,t) = +c\frac{\partial}{\partial y} H_z(x,y,t), \tag{3.63}$$

$$E_x(x,y,t) = \frac{1}{\varepsilon(x,y)} D_x(x,y,t), \tag{3.64}$$

$$\frac{\partial}{\partial t} D_y(x,y,t) = -c \frac{\partial}{\partial x} H_z(x,y,t), \tag{3.65}$$

$$E_y(x,y,t) = \frac{1}{\varepsilon(x,y)} D_y(x,y,t). \tag{3.66}$$

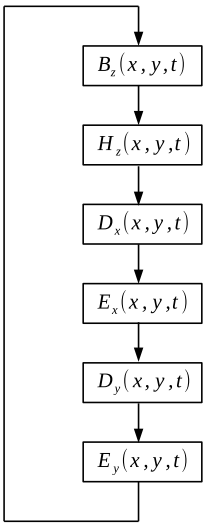

Figura 3.10: Algoritmo de actualización de campos

3.4.2. La Discretización de las Ecuaciones Diferenciales

Se discretiza la primer ecuación en el punto $(x,y,t) = (\Delta x, \Delta y, \Delta t)$

$$\frac{B_z^{n+1/2}(i,j) - B_z^{n-1/2}(i,j)}{\Delta t} = -c \frac{E_y^n(i+1/2,j) - E_y^n(i-1/2,j)}{\Delta x} + c \frac{E_x^n(i,j+1/2) - E_x^n(i,j-1/2)}{\Delta y} \tag{3.67}$$

Una ecuación rotacional se discretiza en el punto $(x,y,t) = [i\Delta x, (j+1/2)\Delta y, (n+1/2)\Delta t]$

$$\frac{D_x^{n+1}(i,j+1/2) - D_x^n(i,j+1/2)}{\Delta t} = +c \frac{H_z^{n+1/2}(i,j+1) - H_z^{n+1/2}(i,j)}{\Delta y} \tag{3.68}$$

La otra ecuación rotacional se discretiza en el punto $(x,y,t) = [(i+1/2)\Delta x, j\Delta y, (n+1/2)\Delta t]$

$$\frac{D_y^{n+1}(i+1/2,j) - D_y^n(i+1/2,j)}{\Delta t} = -c \frac{H_z^{n+1/2}(i+1,j) - H_z^{n+1/2}(i,j)}{\Delta x} \tag{3.69}$$

La actualización de los campos se plantea como un algoritmo (ver Figura 3.10) de la siguiente manera

$$B_z^{n+1/2}(i,j) = B_z^{n-1/2}(i,j) - \frac{1}{2}[E_y^n(i+1/2,j) - E_y^n(i-1/2,j)] + \frac{1}{2}[E_x^n(i,j+1/2) - E_x^n(i,j-1/2)], \tag{3.70}$$

$$H_z^{n+1/2}(i,j) = \frac{1}{\mu(i,j)} B_z^{n+1/2}(i,j), \tag{3.71}$$

$$D_x^{n+1}(i,j+1/2) = D_x^n(i,j+1/2) + \frac{1}{2}[H_z^{n+1/2}(i,j+1) - H_z^{n+1/2}(i,j)], \tag{3.72}$$

$$E_x^{n+1}(i,j+1/2) = \frac{1}{\varepsilon(i,j+1/2)} D_x^{n+1}(i,j+1/2), \tag{3.73}$$

$$D_y^{n+1}(i+1/2,j) = D_y^n(i+1/2,j) - \frac{1}{2}[H_z^{n+1/2}(i+1,j) - H_z^{n+1/2}(i,j)], \tag{3.74}$$

$$E_y^{n+1}(i+1/2,j) = \frac{1}{\varepsilon(i+1/2)} D_y^{n+1}(i+1/2,j). \tag{3.75}$$

3.4.3. La Programación de las Ecuaciones Discretas

Se programan las ecuaciones discretas en dos dimensiones definiendo una malla de simulación de tamaño $100x100$ constituida por aire. Se programa una fuente gaussiana localizada en el punto $x = 50$ y $y = 50$. La fuente genera un pulso que se propaga por 100 periodos de tiempo. El listado del programa se muestra a continuación

Listado 3.3: Ecuaciones discretas en dos dimensiones (te)

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <stdlib.h>
4  #include <iostream>
5  #include <fstream>
6
7  using namespace std;
8
9  #define Nx 100
10 #define Ny 100
11 #define Nt 100
12
13 int main() {
14     int i, j, n;
15     float pi, c;
16     float l0, tao0, w0;
17     float dx, dy, dt;
18     float t, t0, sigma;
19     float Bz[Nx][Ny], Hz[Nx][Ny];
```

```
20      float Ex[Nx][Ny], Dx[Nx][Ny];
21      float Ey[Nx][Ny], Dy[Nx][Ny];
22      float mu[Nx][Ny], epsi_x[Nx][Ny], epsi_y[Nx][Ny];
23
24      pi = 2.0 * asin(1.0);
25      c = 3.0e8;
26
27      l0 = 630.0e9;
28      tao0 = l0 / c;
29      w0 = (2.0 * pi) / tao0;
30
31      t0 = 2.0 * tao0;
32      sigma = 0.5 * tao0;
33
34      dx = l0 / 10.0;
35      dy = dx;
36      dt = dx / (2.0 * c);
37
38      for (i = 0; i < Nx; i++) {
39          for (j = 0; j < Ny; j++) {
40              Bz[i][j] = 0.0;
41              Hz[i][j] = 0.0;
42              Dx[i][j] = 0.0;
43              Ex[i][j] = 0.0;
44              Dy[i][j] = 0.0;
45              Ey[i][j] = 0.0;
46              mu[i][j] = 1.0;
47              epsi_x[i][j] = 1.0;
48              epsi_y[i][j] = 1.0;
49          }
50      }
51
52      for (n = 0; n < Nt; n++) {
53          t = n * dt;
54          for (i = 0; i < Nx; i++) {
55              for (j = 0; j < Ny; j++) {
56                  Bz[i][j] = Bz[i][j] - 0.5 * (Ey[i][j] - Ey[i - 1][j])
57                          + 0.5 * (Ex[i][j] - Ex[i][j - 1]);
58                  Hz[i][j] = (1.0 / mu[i][j]) * Bz[i][j];
59              }
60          }
61
62          Hz[Nx / 2][Ny / 2] = Hz[Nx / 2][Ny / 2]
63                  + exp(-0.5 * ((t - t0) / sigma) * ((t - t0) / sigma));
64          cout << n << " " << Hz[Nx / 2][Ny / 2] << " "
65                  << exp(-0.5 * ((t - t0) / sigma) * ((t - t0) / sigma)) << "\n";
66
67          for (i = 0; i < Nx; i++) {
68              for (j = 0; j < Ny; j++) {
69                  Dx[i][j] = Dx[i][j] + 0.5 * (Hz[i][j + 1] - Hz[i][j]);
70                  Ex[i][j] = (1.0 / epsi_x[i][j]) * Dx[i][j];
71              }
72          }
73
74          for (i = 0; i < Nx; i++) {
75              for (j = 0; j < Ny; j++) {
76                  Dy[i][j] = Dy[i][j] - 0.5 * (Hz[i + 1][j] - Hz[i][j]);
77                  Ey[i][j] = (1.0 / epsi_y[i][j]) * Dy[i][j];
78              }
```

```
79          }
80      } // ciclo temporal
81
82      ofstream myfile;
83      myfile.open("data.dat");
84      for (i = 0; i < Nx; i++) {
85          for (j = 0; j < Ny; j++) {
86              myfile << i << "  " << j << "    " << Hz[i][j] << "\n";
87          }
88      }
89      myfile.close();
90
91      return 0;
92      cout << pi << "\n";
93  }
```

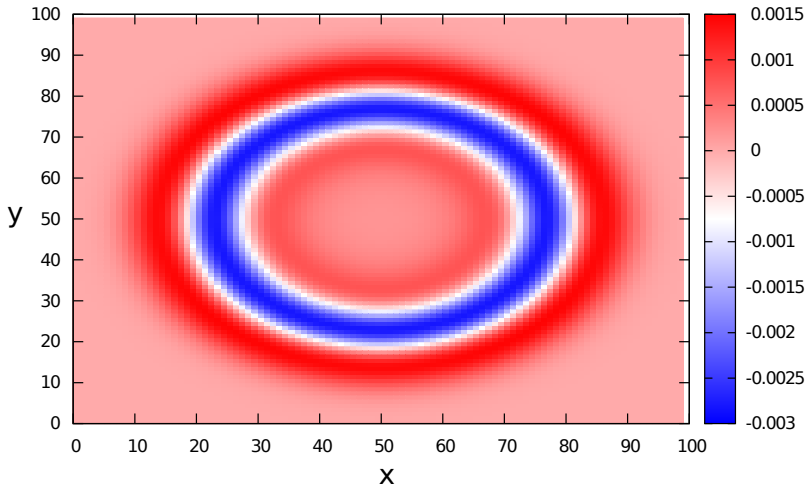

Figura 3.11: Campo magnético en dos dimensiones

3.5. Conclusiones

El método FDTD y sus avances promueven inmensamente el desarrollo del campo del electromagnetismo computacional y juega un papel cada vez más fundamental en las aplicaciones de la nanotecnología. Debido a sus características de extrema flexibilidad y relativamente fácil implementación, el método FDTD es una herramienta indispensable para modelar diversos sistemas y materiales.

En cuanto a posibles líneas de investigación futuras, se espera que el método FDTD sea ampliamente adoptado para simular múltiples fenómenos físicos, a través del cual las propiedades ópticas, eléctricas, térmicas, mecánicas y cuánticas de componentes y dispositivos puedan ser investigadas.

3.6. Referencias

[1] Kane Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14, 1966.

[2] M.E. Taflove, A.; Brodwin. Numerical solution of steady-state electromagnetic scattering problems using the time-dependent maxwell's equations. *IEEE Transactions on Microwave Theory and Techniques*, 23, 1975.

[3] M.E. Taflove, A.; Brodwin. Computation of the electromagnetic fields and induced temperatures within a model of the microwave-irradiated human eye. *IEEE Transactions on Microwave Theory and Techniques*, 23, 1975.

[4] Richard Holland. Threde: A free-field emp coupling and scattering code. *IEEE Transactions on Nuclear Science*, 24, 1977.

[5] Conrad L. Longmire. State of the art in iemp and sgemp calculations. *IEEE Transactions on Nuclear Science*, 22, 1975.

[6] Allen Taflove and Susan C. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method, Third Edition.* Artech House, 2005.

[7] Louis Leithold and Leithold. *Calculus and Analytical Geometry.* Harpercollins College Div, 1997.

[8] J. Manzanares-Martinez, D. Moctezuma-Enriquez, Y. J. Rodriguez-Viveros, B. Manzanares-Martinez, and P. Castro-Garay. Non-perpendicular hypersonic and optical stop-bands in porous silicon multilayers. *Appl. Phys. Lett.*, 101(26):261902, 2012.

[9] Shanhui Fan, JD Joannopoulos, Joshua N Winn, Adrian Devenyi, JC Chen, and Robert D Meade. Guided and defect modes in periodic dielectric waveguides. *JOSA B*, 12(7):1267–1272, 1995.

4 Phoxonics

Con el continuo crecimiento de la potencia de cómputo, el modelado y la simulación numérica han crecido inmensamente como herramienta para entender y analizar prácticamente cualquier problema de ciencia. A mitad del siglo XX, análisis detallados eran requeridos para poder tener una visión significativa de problemas complejos. Hoy en día podemos simplemente conectar las ecuaciones diferenciales con la computadora y obtener una inmensa cantidad de información que desde luego es complementaria a los análisis teóricos [1].

Prácticamente todas las soluciones a problemas en electromagnetismo requieren el uso de una computadora. Incluso cuando se tiene disponible una solución de "forma cerrada" o analítica que es nominalmente exacta, uno normalmente debe utilizar una computadora para traducir esta solución en valores numéricos para un conjunto dado de parámetros [2].

En este capítulo se expone Phoxonics [3] un software que se ha creado para resolver las ecuaciones de Maxwell numéricamente utilizando el método FDTD.

En el tema de cómputo y programación hay una gran variedad de maneras de resolver un problema, usualmente la manera de resolverlo tiene que ver con la flexibilidad y los requerimientos que se quieren tener en el software. A menudo, tal flexibilidad es un factor que suele definir las tecnologías idóneas a utilizar en el desarrollo del software aunque no necesariamente.

A lo largo de la investigación doctoral se realizó un uso extenso de diversas herramientas computacionales para resolver las ecuaciones; sin embargo, cada vez más se acrecentaba la necesidad de encontrar herramientas que cumplieran con todos nuestros requerimientos de cómputo. Después de un tiempo nos dimos cuenta que la mejor manera de cubrir nuestras necesidades de herramientas computacionales era la creación de un software a la medida.

Un diseño flexible permite integrar módulos que pueden resolver las ecuaciones de diferente forma sin necesidad de realizar un nuevo código cada vez. Un mecanismo de configuración permite inicializar e intercambiar módulos de una manera práctica. Hacer uso de estándares permite poder integrar el código actual con otros sistemas así como *entender* formatos de diferentes tipos. Un mecanismo de construcción permite la instanciación eficiente de entidades dinámicamente. Un mecanismo de visualización y estándares para almacenamiento de datos permiten generar una gran cantidad de datos y visualizarlos de diferentes formas.

Como se puede ver, existen mecanismos que nos permiten lograr una arquitectura de software flexible y robusta. Tales mecanismos han sido estudiados y analizados por expertos a lo largo de

los años. Se decidió tomar ventaja de los resultados de los expertos e implementarlos en nuestro software a la medida llamado Phoxonics.

El nombre Phoxonics proviene de la idea de que nuestro software a la medida sea capaz de resolver ecuaciones electromagnéticas así como elásticas. Es decir, photonics y phononics, de ahí la x en Phoxonics. En una primera etapa el software es funcional en la parte de las ecuaciones electromagnéticas y la parte de elasticidad está en progreso. De cualquier manera, los mecanismos y la arquitectura mencionada ya están implementadas en el software.

4.1. Arquitectura

Crear sistemas de cómputo no es una tarea fácil. A medida que la complejidad del sistema aumenta, la tarea de la programación del mismo crece exponencialmente en dificultad [4].

A lo largo de los años se han probado formas de programar los sistemas para mitigar la complejidad a medida que estos crecen, a dichas formas de programar se les llama patrones de diseño. Los patrones de diseño están conformados de pequeños mecanismos que resuelven un problema específico eficientemente. Al conjunto de decisiones que incluyen un grupo de patrones de diseño que se complementan entre sí para conformar un sistema integral se les puede llamar arquitectura.

4.1.1. Lenguaje C++

C++ es un lenguaje vasto y complicado, su sintaxis es a veces intrincada y a menudo es más fácil programar código ineficiente que lo contrario. A pesar de esto, C++ es un lenguaje potente, con una flexibilidad increíble que permite crear código que es rápido y que se puede optimizar a niveles impresionantes. C++ se puede utilizar para resolver casi cualquier problema, desde programación de bajo nivel para dispositivos hasta para sistemas gráficos totalmente orientados a objetos.

C++ es uno de los lenguajes más maduros y utilizados ya que cuenta con más de 30 años de existencia. Existen múltiples librerías creadas en este lenguaje que se pueden utilizar libremente ya que son de código abierto. Las novedades introducidas en C++ en los últimos años hacen que la programación en este lenguaje sea más amigable que nunca [5]. La paralelización con Compute Unified Device Architecture (CUDA) en la programación de Graphics Processor Units (GPUs) es otra de las bondades con que cuenta el lenguaje.

La decisión de utilizar el lenguaje C++ como lenguaje primario recae en el hecho de que el cálculo que implica la resolución del campo electromagnético mediante el método FDTD es muy intenso y la velocidad que C++ puede brindar para resolver este problema es incomparable a la de otros lenguajes.

4.1.2. Sistema Operativo Linux

Linux es un sistema operativo libre, esto significa que no solamente es gratuito sino que también es modificable y su código fuente es accesible sin ningún tipo de restricción. Linux es un sistema robusto, estable y sumamente rápido, es ideal para servidores o para soportar aplicaciones distribuidas que requieren correr en ambientes de alta optimización.

Linux es multitarea y multiusuario, característica con la cual nació, fue pensado de esta manera desde el inicio incluyendo la seguridad con la que cuenta ya que es uno de los sistemas operativos más seguros. Linux cuenta con el mejor sistema de empaquetado de software del mundo y en sus repositorios se puede encontrar una cantidad increíble de software, desde editores de texto, editores de programación hasta librerías altamente optimizadas de ciencia y librerías numéricas.

Linux cuenta con un sistema de actualizaciones muy amigable y fácil de entender. Soporta múltiples arquitecturas de hardware y cuenta con una comunidad enorme de individuos que constantemente están desarrollando nuevas aplicaciones.

Se tomó la decisión de utilizar Linux como sistema operativo principal y específicamente Debian por sus numerosos beneficios. La distribución Debian se ha elegido debido a que se necesita un sistema operativo muy estable y seguro, con actualizaciones de seguridad liberadas regularmente y que no dependa de ninguna organización comercial. También, debido a que cuenta con una comunidad grande de programadores la cual puede dar soporte oportuno a los problemas que puedan existir, y debido a que tiene una robustez y madurez importante [6].

4.1.3. Estándar JSON

Javascript Object Notation (JSON) llamado así por sus siglas en inglés, es un formato ligero de intercambio de datos fácil de leer y escribir para los seres humanos y fácil de analizar y generar para las máquinas. Se basa en un subconjunto del lenguaje de programación JavaScript estándar ECMA-262 [7].

4.1.4. Orientación a Objetos

Un programa orientado a objetos funciona mediante la creación de objetos, conectándolos entre sí y haciendo que colaboren mediante el envío de mensajes. Pero, ¿quién inicializa el proceso? ¿Quién crea el primer objeto y envía el primer mensaje? Todos los programas orientados a objetos tienen un punto de entrada, en éste punto de entrada se ejecutan todas las instrucciones secuencialmente una tras otra.

Cada instrucción puede crear un objeto, conectar objetos o enviar un mensaje a un objeto. Cuando un objeto envía un mensaje, el objeto que recibe el mensaje ejecuta una operación, esta operación también puede crear un objeto, conectar objetos o enviar un mensaje a otro objeto. Por lo tanto, con este mecanismo se satisface el flujo de un programa para lograr cualquier objetivo.

La programación orientada a objetos consta de tres pilares fundamentales con los que se puede construir un diseño, estos pilares son los siguientes:

1. **Encapsulación:** La encapsulación se refiere a un objeto que oculta sus atributos detrás de sus operaciones (sella los atributos en una cápsula, con operaciones expuestas). Los atributos ocultos se dice que son privados.
2. **Herencia:** La herencia nos permite especificar que una clase recibe algunas de sus características a partir de una clase padre y luego añade sus propias características, esto lleva a la descripción de familias enteras de objetos. La herencia nos permite agrupar clases en más y más conceptos generales, por lo que se puede razonar en términos de trozos más grandes del mundo en el que vivimos. Desde el punto de vista de programación queremos herencia porque es compatible con un modelado más rico y más poderoso, esto beneficia tanto al equipo de desarrollo y a otros desarrolladores que quieran reutilizar el código, permite definir la información y el comportamiento en una clase y compartirla con subclases relacionadas. Esto significa que menos código debe ser escrito para lograr el mismo objetivo, es natural, una subclase hereda todos los campos, los mensajes, los métodos (y afirmaciones) de su superclase.
3. **Polimorfismo:** El polimorfismo se deriva de la palabra griega poli, que significa muchos, y morfo que significa forma. Por lo tanto, los medios polimórficos *tienen muchas formas*. El término se puede aplicar por separado a las variables y a los mensajes, una variable polimórfica se refiere a diferentes tipos de valor en diferentes momentos, un mensaje polimórfico significa que se tiene más de un método asociado con él [8].

Phoxonics está diseñado utilizando programación orientada a objetos en donde cada parte del sistema se representa por medio de entidades las cuales colaboran entre sí para lograr objetivos. Esta forma de diseño es utilizada debido a que la orientación a objetos brinda una flexibilidad que de otro modo no se puede obtener. También, debido a la posibilidad de manejar abstracciones las cuales se describen a continuación.

4.2. Abstracciones

En programación orientada a objetos, una abstracción es la definición de un objeto padre que puede contener código reutilizable y del que se pueden derivar uno o muchos objetos hijo. Cada tipo de objeto derivado hereda todo el código del objeto padre, también es posible que el objeto hijo implemente sus propias rutinas de código.

En Phoxonics a cada entidad le corresponde una abstracción. Cada tipo de abstracción contiene métodos comunes para todos los objetos derivados. Por ejemplo, se puede tener diferentes tipos de engines o motores de cálculo, se puede calcular el campo electromagnético utilizando las ecuaciones de Maxwell pero también se puede construir otro componente que utilice ecuaciones distintas.

Ambos tipos de motores compartirían el mismo código definido en la abstracción base, esto nos permite manipular todos los motores que se tengan por medio de la misma abstracción.

Este diseño concuerda con la analogía de los bloques lego donde toda la programación está contenida en clases a las cuales podemos llamar bloques que son intercambiables y configurables. Phoxonics hace un uso intenso de esta técnica para construir simulaciones de manera flexible.

4.3. Entidades

Una entidad es una representación de un objeto o concepto del mundo real que se describe en un modelo de datos, las entidades que contiene Phoxonics son las siguientes.

4.3.1. Entidad Simulation

Representa una simulación y contiene todas las entidades que requieren interactuar entre sí para lograr generar por medio de una configuración una simulación. Las simulaciones se derivan de la abstracción SimulationBase la cual a su vez contiene las siguientes abstracciones: GridBase, SourceBase, DetectorBase, PmlBase, GeometryBase y EngineBase. Estas abstracciones son utilizadas por SimulationBase para construir los bloques de la simulación que están contenidos en la configuración.

La abstracción SimulationBase contiene tres métodos: el método *configure()* que permite configurar la simulación utilizando las abstracciones antes mencionadas. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual. El método *start()* que es el que se encarga de iniciar el proceso de simulación.

La abstracción SimulationBase contiene dos tipos de implementaciones concretas que son: ElectroMagSimulation1D y ElectroMagSimulation2D que representan simulaciones electromagnéticas en una y dos dimensiones respectivamente.

Listado 4.1: Simulation base

```
1  /*
2   * SimulationBase.hpp
3   *
4   *  Created on: Oct 9, 2014
5   *      Author: nano
6   */
7
8  #ifndef SIMULATIONBASE_HPP_
9  #define SIMULATIONBASE_HPP_
10
11 #include "../../../common/common.hpp"
12 #include "../common/SimulationItemBase.hpp"
13 #include "../config/SimulationConfig.hpp"
14 #include "../source/SourceBase.hpp"
15 #include "../source/SourceFactory.hpp"
16 #include "../detector/DetectorBase.hpp"
17 #include "../detector/DetectorFactory.hpp"
```

```
18 #include "../pml/PmlBase.hpp"
19 #include "../pml/PmlFactory.hpp"
20 #include "../geometry/GeometryBase.hpp"
21 #include "../geometry/GeometryFactory.hpp"
22 #include "../engine/EngineBase.hpp"
23 #include "../engine/EngineFactory.hpp"
24 #include "../grid/GridBase.hpp"
25 #include "../grid/GridFactory.hpp"
26 #include "../cell/ElectroMagCell.hpp"
27 #include "../cells/CellsBase.hpp"
28 #include "../cells/CellsFactory.hpp"
29
30 #include <string>
31 #include <memory>
32 #include <iostream>
33 #include <stdexcept>
34 #include <vector>
35 #include <functional>
36
37 namespace phoxonics {
38 namespace core {
39
40 class SimulationBase : public SimulationItemBase {
41 public:
42     // construction / destruction methods
43     explicit SimulationBase();
44     explicit SimulationBase(const std::shared_ptr<SimulationConfig> sim_config);
45     virtual ~SimulationBase();
46
47     std::string simulation_uuid { "" };
48     int simulation_dimensions { 0 };
49     std::string mode { "" };
50
51     std::shared_ptr<GridBase> grid_base;
52     std::vector<std::shared_ptr<SourceBase>> sources_base;
53     std::vector<std::shared_ptr<DetectorBase>> detectors_base;
54     std::shared_ptr<PmlBase> pml_base;
55     std::vector<std::shared_ptr<GeometryBase>> geometries_base;
56     std::shared_ptr<EngineBase> engine_base;
57
58     // configures object and initializes data from config
59     virtual void configure() override;
60
61     // prints object data
62     virtual void print_me() override;
63
64     // prints object data
65     void start();
66
67 private:
68     // construct and configure simulation grid
69     std::shared_ptr<GridBase> build_grid();
70
71     // construct and configure sources
72     std::vector<std::shared_ptr<SourceBase>> build_sources();
73
74     // construct and configure fluxes
75     std::vector<std::shared_ptr<DetectorBase>> build_detectors();
76
```

```
77      // construct and configure pmls
78      std::shared_ptr<PmlBase> build_pml();
79
80      // construct and configure geometries
81      std::vector<std::shared_ptr<GeometryBase>> build_geometries();
82
83      // construct and configure geometries
84      std::shared_ptr<EngineBase> build_engine();
85  };
86
87  } /* namespace core */
88  } /* namespace phoxonics */
89
90  #endif /* SIMULATIONBASE_HPP_ */
```

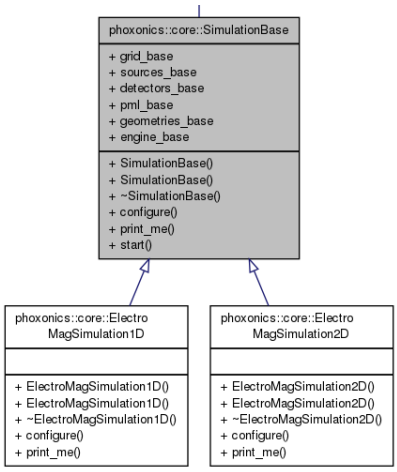

Figura 4.1: Entidad simulation

4.3.2. Entidad Grid

Representa una malla de simulación la cual a su vez contiene celdas. La malla de simulación puede ser de una, dos o tres dimensiones y es el espacio en el cual la simulación se desarrolla. Las mallas se derivan de la abstracción GridBase y las celdas se derivan de CellsBase. La abstracción GridBase contiene un atributo en un tipo de datos de tres dimensiones que define el tamaño de la malla en la simulación. En el tamaño definido de la malla se incluye la capa de absorción Perfectly Matched Layer (PML).

La abstracción GridBase contiene tres métodos: el método *build()* el cual se encarga de construir la malla de simulación con abstracciones de celdas. El método *configure()* el cual se encarga de configurar los bloques de tipo grid de acuerdo a la configuración de la simulación. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual.

La abstracción GridBase contiene dos tipos de implementaciones concretas que son: ElectroMagGrid1D y ElectroMagGrid2D que representan mallas de simulación electromagnéticas en una y dos dimensiones respectivamente.

Listado 4.2: Grid base

```
/*
 * GridBase.hpp
 *
 *  Created on: Jun 25, 2014
 *      Author: nano
 */

#ifndef GRIDBASE_HPP_
#define GRIDBASE_HPP_

#include "../../../common/common.hpp"
#include "../common/SimulationItemBase.hpp"
#include "../config/SimulationConfig.hpp"
#include "../common/Vector3D.hpp"
#include "../cells/CellsBase.hpp"
#include "../cells/CellsFactory.hpp"
#include "../source/SourceBase.hpp"
#include "../pml/PmlBase.hpp"

#include <vector>
#include <memory>

namespace phoxonics {
namespace core {

class GridBase : public SimulationItemBase {
public:
    explicit GridBase();
    explicit GridBase(std::shared_ptr<SimulationConfig> sim_config);
    virtual ~GridBase();

    // vector of cells
    std::shared_ptr<CellsBase> cells_base;
    Vector3D size;

    // Initialize grid
    virtual void build();

    // configures object and initializes data from config
    virtual void configure() override;

    // prints object data
    virtual void print_me() override;
};

} /* namespace core */
} /* namespace phoxonics */

#endif /* GRIDBASE_HPP_ */
```

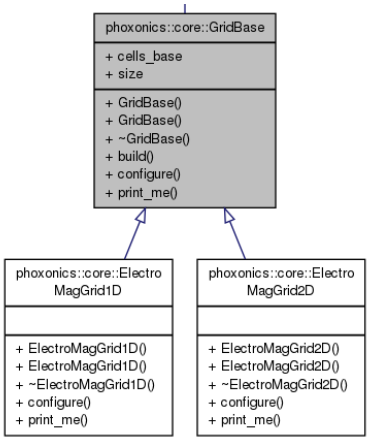

Figura 4.2: Entidad grid

4.3.3. Entidad Cells

Representa un grupo de celdas en la malla de simulación. El contenido total de celdas en una malla depende del tamaño definido en la configuración y puede ser un grupo de una, dos o tres dimensiones, las celdas se derivan de la abstracción CellsBase.

La abstracción CellsBase contiene cinco métodos: el método *configure()* el cual se encarga de configurar los bloques de tipo cells de acuerdo a la configuración de la simulación. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual. El método *generate_cells_h5()* el cual se encarga de generar un archivo HDF5 a partir del grupo de celdas configurado. El método *generate_cells_slice_h5()* el cual se encarga de generar un archivo HDF5 y agregar una capa de datos a partir del grupo de celdas. El método *visualize_cells_slice()* el cual se utiliza para visualizar un grupo de celdas en el caso de que se esté ejecutando la simulación en tiempo real.

La abstracción CellsBase contiene dos tipos de implementaciones concretas que son: ElectroMagCells1D y ElectroMagCells2D que representan los grupos de celdas en las mallas de simulación electromagnéticas en una y dos dimensiones respectivamente.

Listado 4.3: Cells base

```
1  /*
2   * CellsBase.hpp
3   *
4   *  Created on: Jun 25, 2014
5   *      Author: nano
6   */
7
8  #ifndef CELLSBASE_HPP_
9  #define CELLSBASE_HPP_
10
11 #include "../common/SimulationItemBase.hpp"
12 #include "../../../visual/visual.hpp"
13 #include "../../../common/common.hpp"
```

```
14
15 #include <memory>
16
17 namespace phoxonics {
18 namespace core {
19
20 class CellsBase : public SimulationItemBase {
21 public:
22     explicit CellsBase();
23     virtual ~CellsBase();
24
25     virtual void configure() override;
26     virtual void print_me() override;
27     virtual void generate_cells_h5(std::string material_property);
28     virtual void generate_cells_slice_h5(std::string component,
29             std::shared_ptr<phoxonics::common::Hdf5Base> hdf5_base);
30     virtual void visualize_cells_slice(std::string component,
31             phoxonics::visual::GnuplotConfig gp_config);
32 };
33
34 } /* namespace core */
35 } /* namespace phoxonics */
36
37 #endif /* CELLSBASE_HPP_ */
```

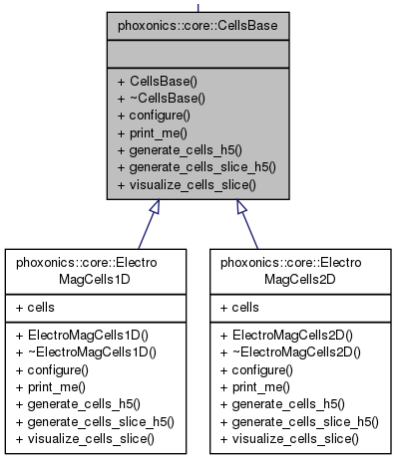

Figura 4.3: Entidad cells

4.3.4. Entidad Material

Representa un tipo de material en la simulación. La entidad material se utiliza en las celdas y en las geometrías para definir de qué tipo de material están constituidos. El material se deriva de la abstracción MaterialBase.

La abstracción MaterialBase no contiene métodos.

La abstracción MaterialBase contiene cuatro tipos de implementaciones concretas que son: Air, Glass, Vacuum y Water que representan los tipos de materiales disponibles en el sistema.

Listado 4.4: Material base

```
/*
 * MaterialBase.hpp
 *
 *  Created on: Nov 12, 2014
 *      Author: nano
 */

#ifndef MATERIALBASE_HPP_
#define MATERIALBASE_HPP_

#include "../common/SimulationItemBase.hpp"
#include <memory>

namespace phoxonics {
namespace core {

class MaterialBase {
public:
    MaterialBase();
    virtual ~MaterialBase();

    // electric properties
    double eps { 0.0 };         // absolute permittivity
    double eps_r { 0.0 };       // relative permittivity (dielectric constant)

    // magnetic properties
    double mu { 0.0 };          // permeability
    double mu_r { 0.0 };        // relative permittivity

    // Electrical resistivity, represented by the Greek letter (rho), is a measure of how
    // strongly a material opposes the flow of electric current. The lower the resistivity,
    // the more readily the material permits the flow of electric charge.
    double rho { 0.0 };

    // Electrical conductivity is the reciprocal quantity of resistivity. Conductivity is a
    // measure of how well a material conducts an electric current. Electric conductivity may
    // be represented by the Greek letter (sigma), (kappa), or (gamma).
    double sigma { 0.0 };
};

} /* namespace core */
} /* namespace phoxonics */

#endif /* MATERIALBASE_HPP_ */
```

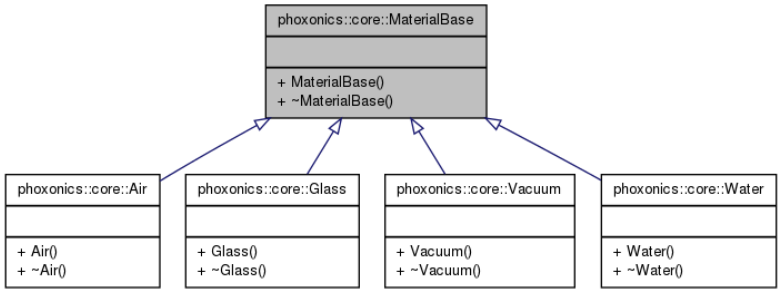

Figura 4.4: Entidad material

4.3.5. Entidad Geometry

Representa una geometría en la simulación. Se pueden definir una cantidad ilimitada de geometrías y cada una puede estar conformada por un material distinto que se define por medio de la entidad material. Las geometrías se derivan de la abstracción GeometryBase.

La abstracción GeometryBase contiene tres atributos: el atributo *location* define la ubicación de la geometría en la malla de simulación. El atributo *size* define el tamaño de la geometría. El atributo *material_type* define el tipo de material del cual está constituida la geometría.

La abstracción GeometryBase contiene cuatro métodos: el método *configure()* el cual se encarga de configurar los bloques de tipo geometry de acuerdo a la configuración de la simulación. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual. El método *generate_geometry()* que se encarga de generar la geometría en la malla de simulación. El método *rotate_geometry_point()* que se encarga de rotar punto por punto una geometría.

La abstracción GeometryBase contiene cuatro tipos de implementaciones concretas que son: Block1D, Block2D, Circle2D y PixelGeometry2D que representan los tipos de geometrías que están disponibles en el sistema.

Listado 4.5: Geometry base

```
1  /*
2   * GeometryBase.hpp
3   *
4   *  Created on: Jun 25, 2014
5   *      Author: nano
6   */
7
8  #ifndef GEOMETRYBASE_HPP_
9  #define GEOMETRYBASE_HPP_
10
11 #include "../config/SimulationConfig.hpp"
12 #include "../common/SimulationItemBase.hpp"
13 #include "../../../common/common.hpp"
14 #include "../material/MaterialBase.hpp"
15 #include "../material/MaterialFactory.hpp"
16 #include "../grid/GridBase.hpp"
17 #include "../common/Vector3D.hpp"
18 #include <memory>
```

```
19
20 namespace phoxonics {
21 namespace core {
22
23 class GeometryBase : public SimulationItemBase {
24 public:
25     explicit GeometryBase();
26     virtual ~GeometryBase();
27
28     Vector3D location;
29     Vector3D size;
30     int rotate { 0 };  // rotate degrees
31     std::shared_ptr<MaterialBase> material_type;
32
33     // configures object and initializes data from config
34     virtual void configure(std::shared_ptr<ConfigBase> config_base) override;
35
36     // prints object data
37     virtual void print_me() override;
38
39     // generate a specific geometry
40     virtual void generate_geometry(std::shared_ptr<GridBase> grid_base);
41
42     // rotate a geometry point and add it to the geometry vector
43     void rotate_geometry_point(int rot_point_x, int rot_point_y, int x, int y, int angle,
44             std::vector<Vector3D>& rotated_geom);
45 };
46
47 } /* namespace core */
48 } /* namespace phoxonics */
49
50 #endif /* GEOMETRYBASE_HPP_ */
```

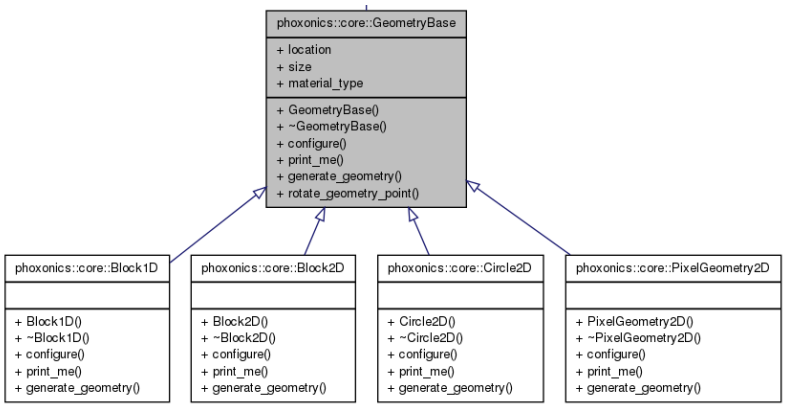

Figura 4.5: Entidad geometry

4.3.6. Entidad Source

Representa una fuente en la simulación. Se pueden definir una cantidad ilimitada de fuentes en el sistema en diferentes lugares de la malla de simulación, de distintos tamaños así como de

diferente tipo. Las fuentes se derivan de la abstracción SourceBase.

La abstracción SourceBase contiene dos atributos: el atributo *location* que define la ubicación de la fuente en la malla de simulación. El atributo *size* que define el tamaño de la fuente en la malla.

La abstracción SourceBase contiene tres métodos: el método *configure()* el cual se encarga de configurar los bloques de tipo source de acuerdo a la configuración de la simulación. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual. El método *apply_source()* que se encarga de hacer el cálculo de la fuente de acuerdo a su tipo y aplicarlo a la simulación.

La abstracción SourceBase contiene seis tipos de implementaciones concretas que son: ElectroMagGaussian1D, ElectroMagGaussian2D, ElectroMagSinusoidal1D, ElectroMagSinusoidal2D, ElectroMagGaussianSin1D y ElectroMagGaussianSin2D que representan los tipos de fuentes que están disponibles en el sistema.

Listado 4.6: Source base

```
1  /*
2   * SourceBase.hpp
3   *
4   *  Created on: Jun 25, 2014
5   *      Author: nano
6   */
7
8  #ifndef SOURCEBASE_HPP_
9  #define SOURCEBASE_HPP_
10
11 #include "../config/SimulationConfig.hpp"
12 #include "../common/SimulationItemBase.hpp"
13 #include "../common/Vector3D.hpp"
14 #include "../cells/CellsBase.hpp"
15 #include "../../../common/common.hpp"
16 #include <string>
17 #include <memory>
18
19 namespace phoxonics {
20 namespace core {
21
22 class SourceBase : public SimulationItemBase {
23 public:
24     explicit SourceBase();
25     virtual ~SourceBase();
26
27     std::string component { "" };
28     Vector3D location;
29     Vector3D size;
30     std::string source_type { "" };
31
32     // configures object and initializes data from config
33     virtual void configure(std::shared_ptr<ConfigBase> config_base) override;
34
35     // prints object data
36     virtual void print_me() override;
37
```

```
38      // apply the source to the grid cells
39      virtual void apply_source(std::shared_ptr<CellsBase> cells_base, double time);
40  };
41
42  } /* namespace core */
43  } /* namespace phoxonics */
44
45  #endif /* SOURCEBASE_HPP_ */
```

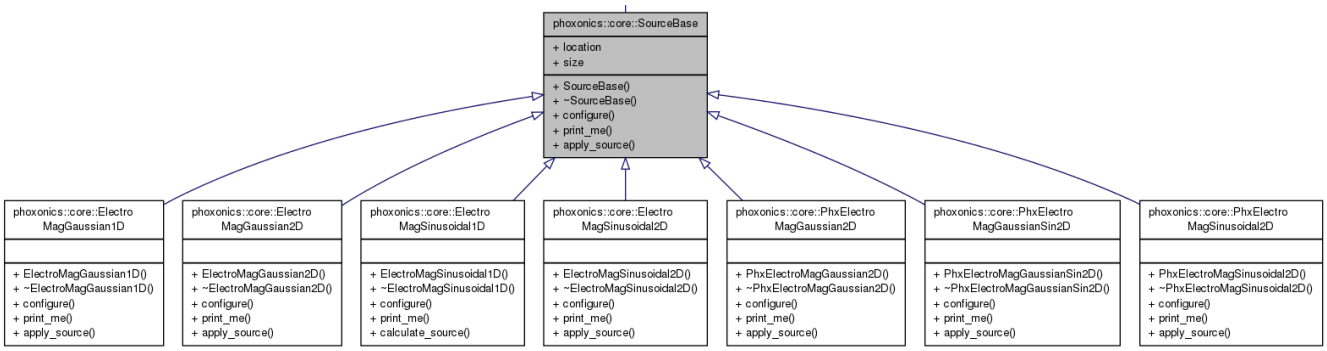

Figura 4.6: Entidad source

4.3.7. Entidad Detector

Representa un detector en la simulación. Se pueden definir una cantidad ilimitada de detectores en el sistema en diferentes lugares de la malla de simulación y de diferentes tamaños. Los detectores se derivan de la abstracción DetectorBase.

La abstracción DetectorBase contiene dos atributos: el atributo *location* que define la ubicación del detector en la malla de simulación. El atributo *size* que define el tamaño del detector en la malla.

La abstracción DetectorBase contiene seis métodos: el método *configure()* el cual se encarga de configurar los bloques de tipo detector de acuerdo a la configuración de la simulación. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual. El método *init_detector()* que se encarga de inicializar los datos internos del detector. El método *apply_detector()* que se encarga de procesar los datos del detector actual. El método *calculate_amplitude_phase()* que se encarga de calcular la amplitud y la fase por medio de la transformada de Fourier. El método *record_hdf5_detector()* que se encarga de grabar los datos del detector a un archivo de tipo HDF5.

La abstracción DetectorBase contiene dos tipos de implementaciones concretas que son: ElectroMagDetector1D y ElectroMagDetector2D, que representan los tipos de detectores que están disponibles en el sistema.

Listado 4.7: Detector base

```
1  /*
```

```
 * DetectorBase.hpp
 *
 *  Created on: Dec 2, 2014
 *      Author: nano
 */

#ifndef DETECTORBASE_HPP_
#define DETECTORBASE_HPP_

#include "../config/SimulationConfig.hpp"
#include "../common/SimulationItemBase.hpp"
#include "../cells/CellsBase.hpp"
#include "../../../common/common.hpp"
#include "DetectorData.hpp"
#include <memory>
#include <vector>

namespace phoxonics {
namespace core {

class DetectorBase : public SimulationItemBase {
public:
    explicit DetectorBase();
    virtual ~DetectorBase();

    std::string component { "" };
    Vector3D location;
    Vector3D size;
    std::vector<DetectorData> data;
    std::string h5_data_file { "" };
    std::string h5_dataset_name { "" };

    // a configures object and initializes data from config
    virtual void configure(std::shared_ptr<ConfigBase> config_base) override;

    // prints object data
    virtual void print_me() override;

    // initialize detector
    virtual void init_detector();

    // apply detector logic
    virtual void apply_detector(std::shared_ptr<CellsBase> cells_base, double time);

    // calculate phase and amplitude of detector
    virtual void calculate_amplitude_phase();

    // records detector data in hdf5 format
    virtual void record_hdf5_detector();

    std::string h5_coord_data_file();
};

} /* namespace core */
} /* namespace phoxonics */

#endif /* DETECTORBASE_HPP_ */
```

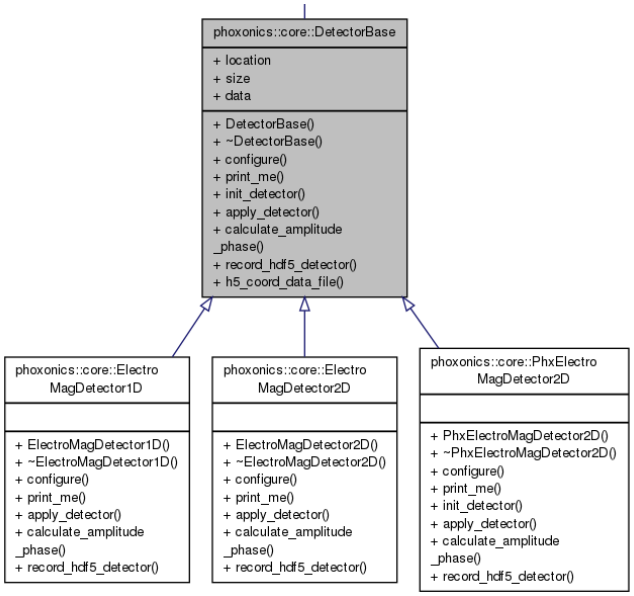

Figura 4.7: Entidad detector

4.3.8. Entidad Pml

Representa la capa de absorción perfecta en la simulación y se deriva de la abstracción PmlBase.

La abstracción PmlBase contiene un atributo: el atributo *thickness* que define el tamaño de la capa de absorción en términos de celdas.

La abstracción PmlBase contiene tres métodos: el método *configure()* el cual se encarga de configurar los bloques de tipo PML de acuerdo a la configuración de la simulación. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual. El método *apply_pml()* que se encarga de aplicar la lógica de la PML en cada paso de la simulación.

La abstracción PmlBase contiene dos tipos de implementaciones concretas que son: ElectroMagPml1D y ElectroMagPml2D, que representan los tipos de PMLs que están disponibles en el sistema.

Listado 4.8: Pml base

```
1  /*
2   * PmlBase.hpp
3   *
4   *  Created on: Jun 25, 2014
5   *      Author: nano
6   */
7
8  #ifndef PMLBASE_HPP_
9  #define PMLBASE_HPP_
10
11 #include "../config/SimulationConfig.hpp"
12 #include "../common/SimulationItemBase.hpp"
13 #include "../common/Vector3D.hpp"
14 #include "../cells/CellsBase.hpp"
```

```
15 #include "../../../common/common.hpp"
16 #include <memory>
17 #include <string>
18
19 namespace phoxonics {
20 namespace core {
21
22 class PmlBase : public SimulationItemBase {
23 public:
24     explicit PmlBase();
25     virtual ~PmlBase();
26
27     Vector3D thickness;
28
29     // configures object and initializes data from config
30     virtual void configure(std::shared_ptr<ConfigBase> config_base) override;
31
32     // prints object data
33     virtual void print_me() override;
34
35     // apply pml
36     virtual void apply_pml(std::shared_ptr<CellsBase> cells_base);
37 };
38
39 } /* namespace core */
40 } /* namespace phoxonics */
41
42 #endif /* PMLBASE_HPP_ */
```

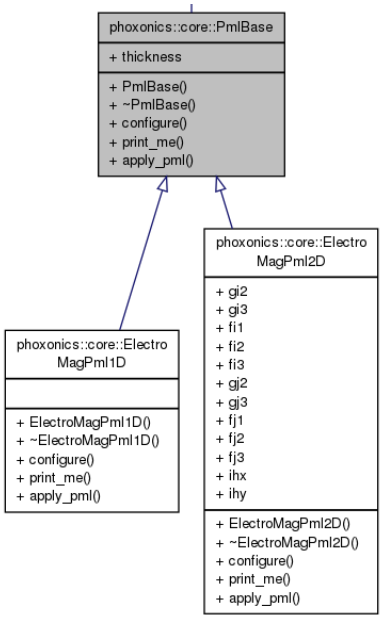

Figura 4.8: Entidad pml

4.3.9. Entidad Engine

Representa el motor de cálculo a utilizar en la simulación. El motor de cálculo puede ser intercambiable al igual que todas las demás entidades. Los engines o motores de cálculo se derivan de la abstracción EngineBase.

La abstracción EngineBase contiene cuatro atributos: el atributo *grid_base_* que define el grid que se esta utilizando en la simulación. El atributo *sources_base_* que define todas las fuentes definidas en la simulación. El atributo *detectors_base_* que define todos los detectores definidos en la simulación. El atributo *pml_base_* que define el PML de la simulación. Estos atributos son utilizados por el motor de cálculo para hacer las operaciones de la simulación.

La abstracción EngineBase contiene catorce métodos: el método *configure()* el cual se encarga de configurar los bloques de tipo engine de acuerdo a la configuración de la simulación. El método *print_me()* que se encarga de imprimir a consola los valores del objeto actual. El método *start()* que se encarga de iniciar el proceso de simulación. El método *process()* que se encarga del proceso principal de la simulación así como de el ciclo de pasos inicial. El método *update_field()* que se encarga de actualizar los campos electromagnéticos en cada paso de la simulación. El método *apply_source()* que se encarga de aplicar cada una de las fuentes definidas en la simulación en cada paso del proceso. El método *apply_pml()* que se encarga de aplicar la PML definida en cada paso de la simulación. El método *apply_detectors()* que se encarga de aplicar todos los detectores definidos en la simulación en cada paso del proceso. El método *init_hdf5_files()* que se encarga de inicializar todos los archivos en formato HDF5 que se utilizan en la simulación. El método *record_hdf5_slice()* que se encarga de grabar un grupo de datos a disco generados en cada paso del proceso de simulación. El método *visualize_real_time()* que se encarga de visualizar los datos de los pasos del proceso de simulación en tiempo real en caso de que esté definido de esta forma en la configuración. El método *calculate_detectors_amp_phase()* que se encarga de calcular la amplitud y la fase de todos los detectores en la simulación. El método *record_hdf5_detectors()* que se encarga de grabar toda la información de los detectores a disco utilizando el formato HDF5. El método *generate_hdf5_shell()* que crea un script en el lenguaje bash que se encarga de generar la animación y los archivos de imagen de la simulación a partir de los datos de la misma que están en formato HDF5.

La abstracción EngineBase contiene dos tipos de implementaciones concretas que son: ElectroMagnetic1D y ElectroMagnetic2D, que representan los tipos de motores de cálculo que están disponibles en el sistema.

Listado 4.9: Engine base

```
1 /*
2  * EngineBase.hpp
3  *
4  *  Created on: Oct 20, 2014
5  *      Author: nano
6  */
7
```

```
8  #ifndef ENGINEBASE_HPP_
9  #define ENGINEBASE_HPP_
10
11 #include "../config/SimulationConfig.hpp"
12 #include "../common/SimulationItemBase.hpp"
13 #include "../grid/GridBase.hpp"
14 #include "../source/SourceBase.hpp"
15 #include "../detector/DetectorBase.hpp"
16 #include "../pml/PmlBase.hpp"
17 #include "../data/EmConstants.hpp"
18 #include "../../../common/common.hpp"
19 #include "../../../visual/visual.hpp"
20 #include <memory>
21 #include <string>
22
23 namespace phoxonics {
24 namespace core {
25
26 class EngineBase : public SimulationItemBase {
27 public:
28     explicit EngineBase();
29     virtual ~EngineBase();
30
31     bool mpi_enabled { false };
32     bool cuda_enabled { false };
33     std::string h5_geometry_file { "" };
34     std::string h5_data_file { "" };
35     std::string h5_geom_dataset_name { "" };
36     std::string h5_data_dataset_name { "" };
37     bool real_time_visualization { false };
38     std::string visualization_commands { "" };
39     bool generate_animation { false };
40
41     // configures object and initializes data from config
42     virtual void configure() override;
43
44     // prints object data
45     virtual void print_me() override;
46
47     // starts engine process
48     virtual void start(std::shared_ptr<GridBase> grid_base,
49             std::vector<std::shared_ptr<SourceBase>> sources_base,
50             std::vector<std::shared_ptr<DetectorBase>> detectors_base,
51             std::shared_ptr<PmlBase> pml_base);
52
53 protected:
54     std::shared_ptr<GridBase> grid_base_;
55     std::vector<std::shared_ptr<SourceBase>> sources_base_;
56     std::vector<std::shared_ptr<DetectorBase>> detectors_base_;
57     std::shared_ptr<PmlBase> pml_base_;
58
59     virtual void process();
60     virtual void update_field(std::string component);
61     virtual void apply_sources(double time);
62     virtual void apply_pml();
63     virtual void apply_detectors(double time);
64     virtual void init_hdf5_files();
65     virtual void record_hdf5_slice();
66     virtual void visualize_real_time();
```

```
67      virtual void calculate_detectors_amp_phase();
68      virtual void record_hdf5_detectors();
69      virtual void generate_hdf5_shell(bool execute_shell);
70      std::string h5_component_data_file(std::string component);
71  };
72
73  } /* namespace core */
74  } /* namespace phoxonics */
75
76  #endif /* ENGINEBASE_HPP_ */
```

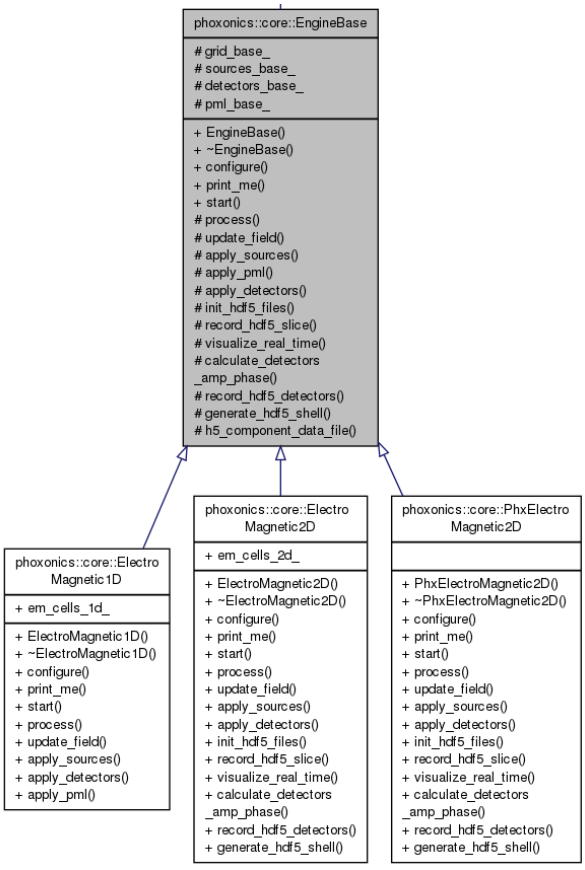

Figura 4.9: Entidad engine

4.4. Mecanismos

En esta sección se describen algunos de los mecanismos que utiliza Phoxonics para manejar la configuración, así como para administrar las simulaciones y la construcción de las mismas.

4.4.1. Configuración

Como se ha mencionado anteriormente Phoxonics utiliza el estándar JSON para manejar la configuración de simulaciones con la intención de poder conectar diferentes tipos de aplicaciones por medio de un formato flexible y fácil de usar. Las configuraciones están divididas en secciones lógicas que se asemejan a las entidades. Cada una de las secciones se derivan de la abstracción ConfigBase.

La abstracción ConfigBase contiene ocho tipos de implementaciones concretas que son: CellConfig, DetectorConfig, EngineConfig, GeometryConfig, GridConfig, PmlConfig, SimulationConfig y SourceConfig, que representan los tipos de configuraciones que están disponibles en el sistema.

Listado 4.10: Config base

```
1  /*
2   * ConfigBase.hpp
3   *
4   *  Created on: Oct 15, 2014
5   *      Author: nano
6   */
7
8  #ifndef CONFIGBASE_HPP_
9  #define CONFIGBASE_HPP_
10
11 #include "../../../common/common.hpp"
12 #include "ConfigParam.hpp"
13
14 #include <string>
15 #include <iostream>
16
17 namespace phoxonics {
18 namespace core {
19
20 class ConfigBase {
21 public:
22     explicit ConfigBase();
23     virtual ~ConfigBase();
24
25     // print object in console
26     virtual void print_me();
27
28     // log class
29     phoxonics::common::Logger log;
30
31     // generic parameters for config
32     std::vector<ConfigParam> config_param;
33
34     // config has been already filled with data
35     bool has_parsed_data { false };
36
37     // is config requiered
38     bool is_required { false };
39
40     // The congif type to resolve dynamically
41     std::string config_type { "" };
42
43     // each configuration block is responsible for providing its description
```

```
44      std::string config_desc { "Base Config" };
45
46      // formatter typedef
47      typedef phoxonics::common::Formatter fmt;
48
49      phoxonics::common::Utilities utils;
50  };
51
52  } /* namespace core */
53  } /* namespace phoxonics */
54
55  #endif /* CONFIGBASE_HPP_ */
```

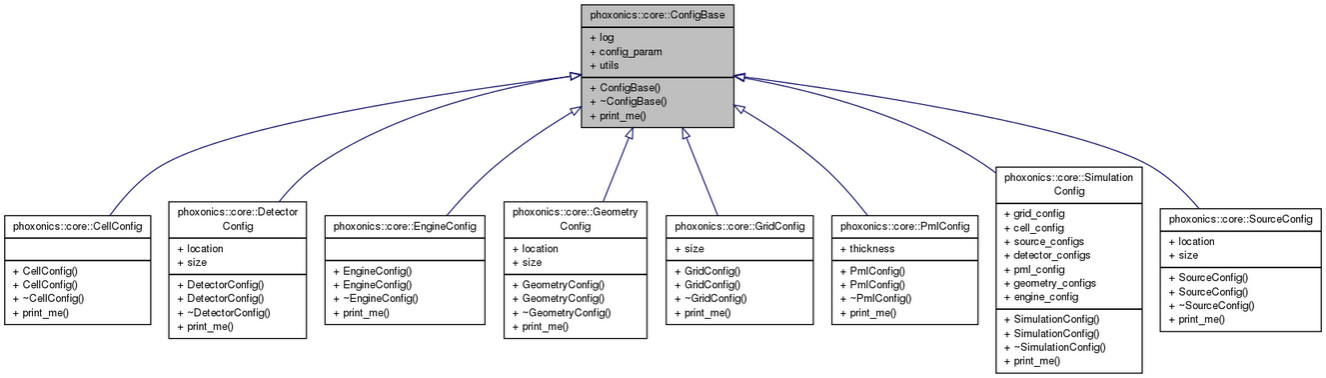

Figura 4.10: Mecanismo de configuración

La simulación que se quiere construir cuenta con un archivo de configuración en formato JSON. El archivo de configuración es leído desde disco y transformado a un objeto en C++ por medio de una clase crítica que se encarga de el manejo de las configuraciones llamada ConfigJsonParser. El objeto ConfigJsonParser inicializa todos los objetos a partir del archivo de configuración de la simulación.

Listado 4.11: Config json parser

```
1  /*
2   * ConfigJsonParser.hpp
3   *
4   *  Created on: Oct 12, 2014
5   *      Author: nano
6   */
7
8  #ifndef CONFIGJSONPARSER_HPP_
9  #define CONFIGJSONPARSER_HPP_
10
11 #include "MainConfig.hpp"
12 #include "LoggerConfig.hpp"
13 #include "SimulationConfig.hpp"
14 #include "ConfigBase.hpp"
15 #include "GridConfig.hpp"
16 #include "CellConfig.hpp"
17 #include "DetectorConfig.hpp"
```

```
18 #include "EngineConfig.hpp"
19 #include "SourceConfig.hpp"
20 #include "PmlConfig.hpp"
21 #include "ConfigParam.hpp"
22 #include "../common/Vector3D.hpp"
23 #include "GeometryConfig.hpp"
24 #include "ConfigFactory.hpp"
25 #include "../common/ConfigParamHelper.hpp"
26
27 #include <fstream>
28 #include <iostream>
29 #include <string>
30 #include <vector>
31 #include <memory>
32 #include <stdexcept>
33
34 #include "../../../cpp-json/include/cpp-json/json.h"
35 #include "../../../common/common.hpp"
36
37 namespace phoxonics {
38 namespace core {
39
40 class ConfigJsonParser {
41 public:
42     // default construction
43     explicit ConfigJsonParser();
44     explicit ConfigJsonParser(std::string json_config_file);
45     virtual ~ConfigJsonParser();
46
47     phoxonics::common::Logger log;
48     std::string json_config_file;
49
50     /************************* Main Confg *************************/
51     // parse configuration json objects
52     MainConfig parse_main_config();
53
54     // parse configuration json objects
55     MainConfig parse_main_config(const json::value& root_json_val);
56
57     // parse logger json objects
58     LoggerConfig parse_logger_config(const json::value& root_json_val);
59     /**************************************************************/
60
61     /********************** Simulations Confg **********************/
62     // parse simulation json objects
63     SimulationConfig parse_simulation_config();
64
65     // parse a simulation json object
66     SimulationConfig parse_simulation_config(const json::value &root_json_val);
67
68     // parse a grid json object
69     GridConfig parse_grid_config(const json::value &root_json_val);
70
71     // parse a cell json object
72     CellConfig parse_cell_config(const json::value &root_json_val);
73
74     // parse a source json objects
75     std::vector<SourceConfig> parse_source_configs(const json::value &root_json_val);
76
```

```
77      // parse a detector json object
78      std::vector<DetectorConfig> parse_detector_configs(const json::value &root_json_val);
79
80      // parse a pml json objects
81      PmlConfig parse_pml_config(const json::value &root_json_val);
82
83      // parse a pml json objects
84      std::vector<GeometryConfig> parse_geometry_configs(const json::value &root_json_val);
85
86      // parse a engine json object
87      EngineConfig parse_engine_config(const json::value &root_json_val);
88
89      // add config parameters for all configs
90      void add_config_params(ConfigBase& config_instance, const json::value& json_val);
91
92      // parse json config param objects
93      std::vector<ConfigParam> parse_config_params(const json::value& json_val);
94
95      // parse json vector 3d objects
96      Vector3D parse_vector_3d(const json::value &json_val);
97
98      // sample supported format for frequencies: "200-205,215,220"
99      std::vector<int> str_freqs_to_vector(std::string str_freqs);
100     /*****************************************************************/
101
102 private:
103     // common utils
104     phoxonics::common::File file_;
105     phoxonics::common::Strings str_;
106     ConfigParamHelper cfg_parm_helper_;
107 };
108
109 } /* namespace core */
110 } /* namespace phoxonics */
111
112 #endif /* CONFIGJSONPARSER_HPP_ */
```

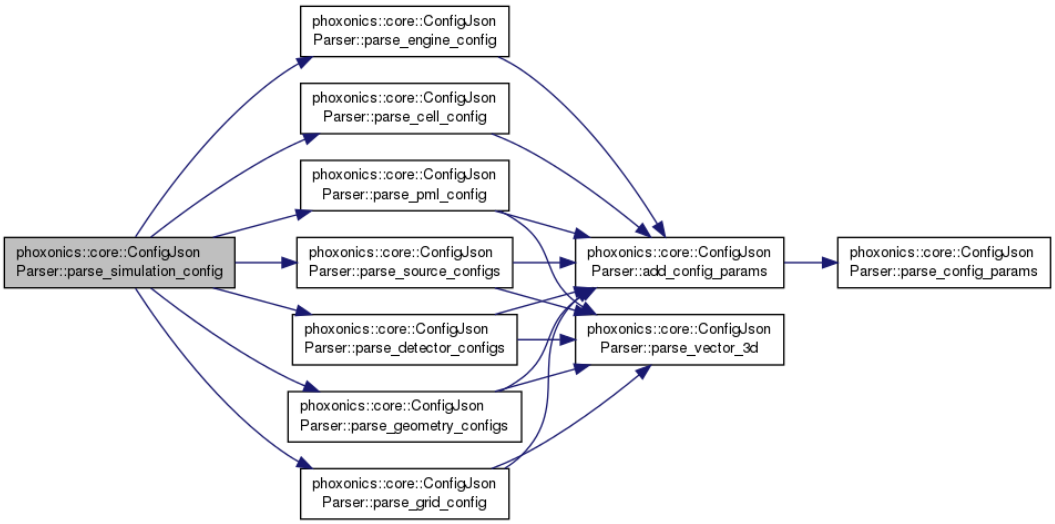

Figura 4.11: Mecanismo de configuración

4.4.2. Administrador de Simulaciones

El administrador de simulaciones consta de una clase que contiene lógica de administración, el nombre de esta clase es SimulationManager.

La clase SimulationManager contiene dos métodos: el método *construct_simulation()* que se encarga de dar la señal para la construcción de la simulación en base a su configuración. El método *start_simulation()* que se encarga de dar la señal para iniciar el proceso de simulación.

Listado 4.12: Simulation manager

```
1  /*
2   * SimulationManager.hpp
3   *
4   *  Created on: Oct 25, 2014
5   *      Author: cuda
6   */
7
8  #ifndef SIMULATIONMANAGER_HPP_
9  #define SIMULATIONMANAGER_HPP_
10
11 #include "../config/ConfigJsonParser.hpp"
12 #include "../config/ConfigValidator.hpp"
13 #include "../../../common/common.hpp"
14 #include "../config/SimulationConfig.hpp"
15 #include "SimulationBase.hpp"
16 #include "SimulationFactory.hpp"
17 #include "SimulationBuilder.hpp"
18 #include "../common/SimulationItemBase.hpp"
19
20 #include <string>
21 #include <memory>
22 #include <iostream>
23 #include <stdexcept>
24
```

```
25 namespace phoxonics {
26 namespace core {
27
28 class SimulationManager : public SimulationItemBase {
29 public:
30     explicit SimulationManager(std::string sims_folder, std::string simulation_file);
31     virtual ~SimulationManager();
32
33     // start a specific simulation
34     virtual void start_simulation();
35
36     // construct all parts of simulation into memory
37     virtual void construct_simulation();
38
39 private:
40     // common utils
41     phoxonics::common::File file_;
42
43     // the main configuration file
44     std::string main_config_file_ { "main_config.json" };
45
46     // the simulation configuration file, it can contain multiple simulations
47     std::string sim_config_file_ { "" };
48
49     // general main config parsed from main config file
50     MainConfig main_config_;
51
52     // list of simulation configs parsed from simulation file
53     SimulationConfig sim_config_;
54
55     // parse config file and configure logger obj
56     void configure_logger();
57
58     // parse simulation config file into obj
59     void parse_simulation_config();
60
61     // construction and configuration helper
62     SimulationBuilder sim_builder_;
63
64     // constructed simulation
65     std::shared_ptr<SimulationBase> sim_base_;
66
67     // config validator
68     ConfigValidator config_validator_;
69 };
70
71 } /* namespace core */
72 } /* namespace phoxonics */
73
74 #endif /* SIMULATIONMANAGER_HPP_ */
```

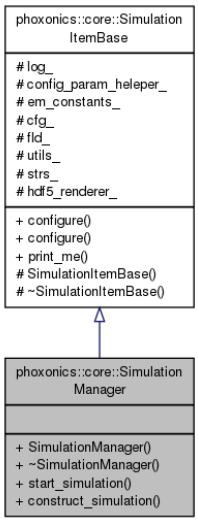

Figura 4.12: Administrador de simulaciones

4.4.3. Construcción

La construcción de una simulación consta de dos partes fundamentales. La primera parte es una clase que se encarga de generar celdas, geometrías y la simulación en general. La segunda parte es la utilización del patron de diseño factory en todas las entidades de la simulación.

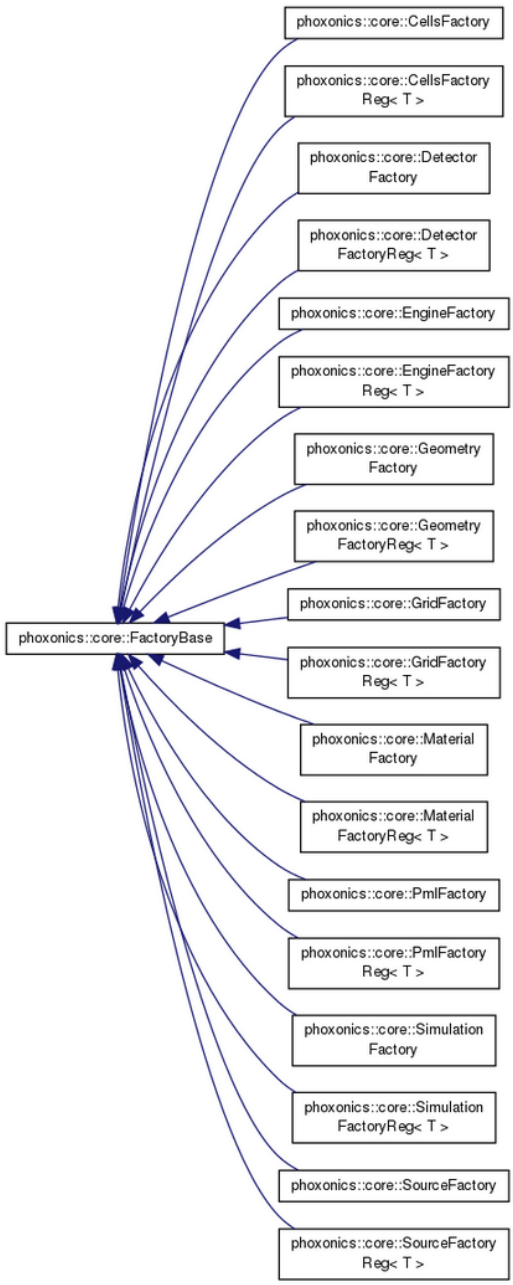

Figura 4.13: Construcción de entidades

Listado 4.13: Simulation builder

```
1 /*
2  * SimulationBuilder.hpp
3  *
4  *  Created on: Nov 14, 2014
5  *      Author: cuda
6  */
7
8 #ifndef SIMULATIONBUILDER_HPP_
```

```cpp
#define SIMULATIONBUILDER_HPP_

#include "SimulationBase.hpp"
#include "SimulationFactory.hpp"
#include "../source/SourceBase.hpp"
#include "../source/SourceFactory.hpp"
#include "../pml/PmlBase.hpp"
#include "../pml/PmlFactory.hpp"
#include "../geometry/GeometryBase.hpp"
#include "../geometry/GeometryFactory.hpp"
#include "../engine/EngineBase.hpp"
#include "../engine/EngineFactory.hpp"
#include "../grid/GridBase.hpp"
#include "../grid/GridFactory.hpp"
#include "../cell/ElectroMagCell.hpp"
#include "../cells/CellsBase.hpp"
#include "../cells/CellsFactory.hpp"
#include "../common/SimulationItemBase.hpp"

#include <string>
#include <memory>
#include <iostream>
#include <stdexcept>

namespace phoxonics {
namespace core {

class SimulationBuilder {
public:
    SimulationBuilder();
    explicit SimulationBuilder(SimulationConfig sim_config);
    virtual ~SimulationBuilder();

    // construct and configure simulation
    std::shared_ptr<SimulationBase> build_simulation();

    // generate geometries in grid
    void generate_geometries(std::shared_ptr<SimulationBase> sim_base);

    // generate h5 file from grid
    void generate_cells_h5(std::shared_ptr<SimulationBase> sim_base,
            std::string material_property);

private:
    // simulation configuration
    SimulationConfig sim_config_;

    phoxonics::common::Logger log_;
    typedef phoxonics::common::Formatter fmt_;
};

} /* namespace core */
} /* namespace phoxonics */

#endif /* SIMULATIONBUILDER_HPP_ */
```

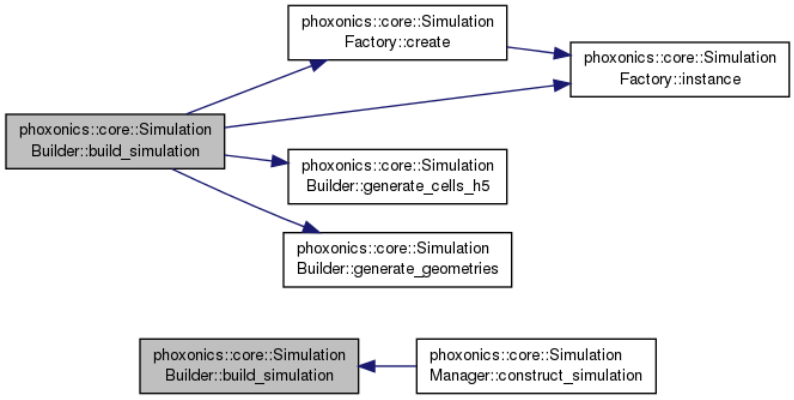

Figura 4.14: Construcción de simulaciones

4.5. Herramientas de Servidor

Para el desarrollo de Phoxonics se han incluido múltiples herramientas al ciclo de programación para facilitar y mejorar la calidad del código, entre las herramientas más importantes se encuentran las siguientes

4.5.1. Integración Continua

El sistema de integración continua o Continuous Integration (CI) por sus siglas en inglés, se refiere a un sistema global que consta de múltiples componentes y cuyo objetivo es integrar todas las partes que conforman un software. El sistema de CI puede forzar reglas, correr tests, correr múltiples herramientas y asegurarse de que todas las partes del software funcionan como una sola unidad. El sistema de CI consta de los siguientes componentes

- **Código cliente:** Se refiere al código o a los sistemas clientes, es decir, a los sistemas de los programadores del software que producen código fuente.
- **Versionador de código:** Se refiere a un sistema centralizado que se encarga de manejar el control de versiones del código fuente (más detalles en la siguiente sección).
- **Servidor de CI:** Se refiere al servidor en donde se encuentra configurado un software de CI.
- **Script de compilación:** Se refiere a todas las instrucciones de código que en conjunto hacen posible la integración de todos los componentes del software final.

Normalmente los pasos a seguir en un escenario de integración continua son los siguientes

1. Un programador sube código al servidor por medio del versionador de código.
2. El software de CI detecta los cambios de código mediante el repositorio de código.
3. El servidor de CI recupera la última copia del código de el repositorio y luego ejecuta el script de construcción que integra el software.

4. El servidor de CI genera retroalimentación por correo electrónico enviando los resultados de construcción a los miembros del proyecto especificados.
5. El servidor de CI continúa consultando para verificar si hay cambios en el repositorio de control [9].

Phoxonics cuenta con un sistema de CI el cual está configurado para correr en una plataforma Linux utilizando la distribución Debian. El software encargado de integrar se llama Jenkins el cual es de código abierto y se puede instalar libremente. El script de construcción o integración tiene varios objetivos como lo son: compilar todas las partes del código del software, asegurarse de que se cumplan las reglas de estándares de código y asegurarse que los tests del sistema se ejecuten y no fallen.

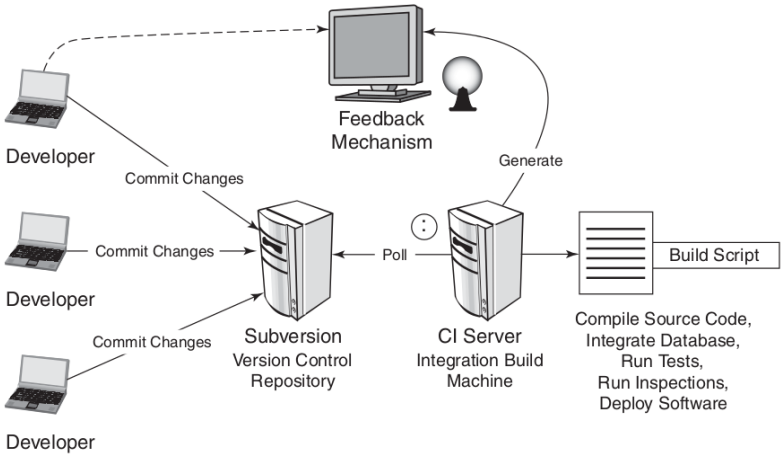

Figura 4.15: Componentes de un sistema de ci

4.5.2. Control de Versiones

El control de versiones es un sistema que registra cambios en un archivo o conjunto de archivos en el tiempo. El control de versiones se utiliza mucho en archivos de texto aunque prácticamente se puede aplicar para cualquier tipo de archivo que se quiera versionar. El control de versiones guarda las modificaciones que se le han hecho a los archivos y las hace disponibles por medio de comandos.

Si se quiere regresar a una versión de un archivo anterior se puede hacer, así como comparar versiones y hacer ajustes entre versiones. Normalmente el control de versiones se ejecuta en un servidor aunque los controles de versiones más robustos son aquellos que se encuentran distribuidos entre las máquinas de los programadores.

En un control de versiones distribuido se cuenta con un repositorio local y un repositorio remoto, primero los cambios se graban localmente y después se hace una sincronización con el servidor para hacer estos cambios disponibles a otros programadores.

Phoxonics utiliza git como control de versiones. Git es un sistema de control de versiones libre que se puede utilizar sin problemas legales en varias plataformas. Git se integra de una manera eficiente en plataformas linux, es muy rápido y flexible y permite configurar un sistema de CI relativamente fácil.

4.6. Front-End

El front-end es la parte gráfica del software en donde el usuario final elige opciones y construye configuraciones de simulaciones fácilmente. Gracias a las decisiones de arquitectura y estándares establecidos en Phoxonics es posible desarrollar una parte gráfica del sistemas en múltiples tecnologías.

La tecnología seleccionada para la parte gráfica es una tecnología web. Se ha decidido hacer uso de tecnologías web debido a la posibilidad de acceder al software por medio de internet. A continuación se mencionan las herramientas utilizadas en el desarrollo del front-end

- **JQuery:** Es una librería que permite la manipulación de elementos contenidos en un documento de navegación.
- **Html5:** Tecnología novedosa que expone nuevos controles y tipos de objetos que son programables y ejecutables en el navegador de internet.
- **Fabric.js:** Framework de JavaScript que expone funciones para controlar geometrías sobre controles de Html5.
- **Php:** Lenguaje de programación del lado del servidor que normalmente se utiliza para darle dinamismo a las páginas web.

A continuación se muestra el ambiente gráfico con el que cuenta Phoxonics

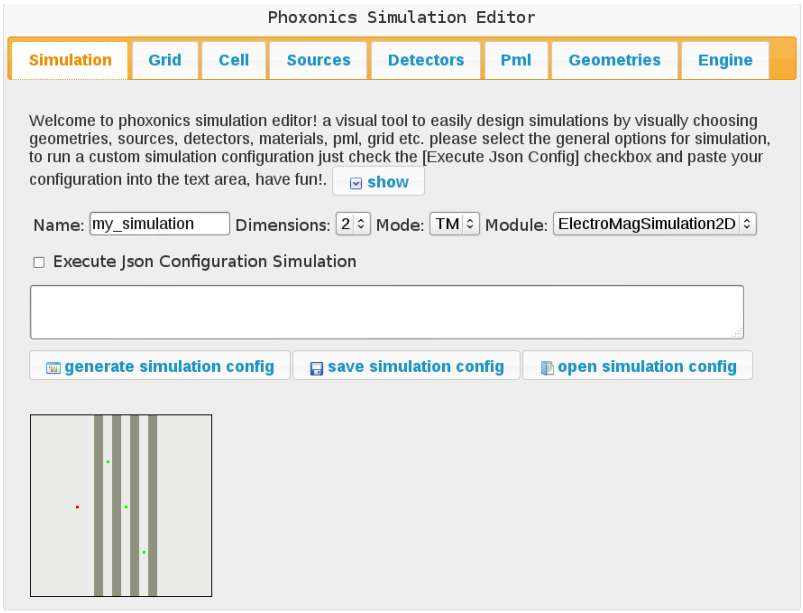

Figura 4.16: Frontend phoxonics - simulation

En la figura 4.16 se muestra la parte general de la simulación donde se configuran las opciones más básicas

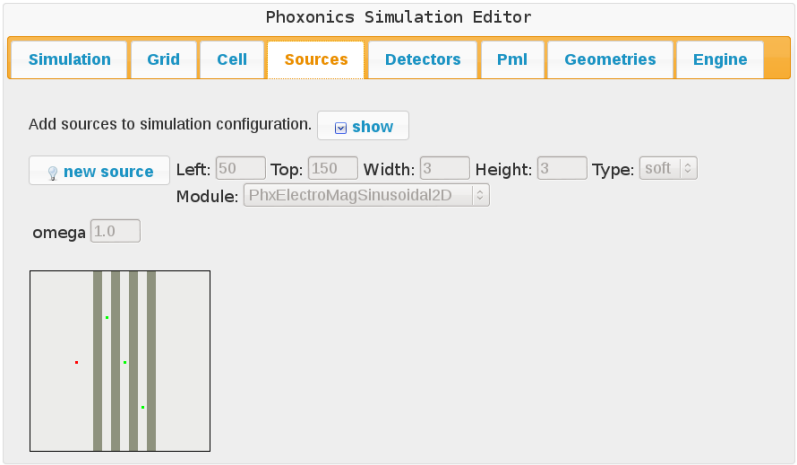

Figura 4.17: Frontend phoxonics - sources

En la figura 4.17 se muestran las fuentes. En esta parte se pueden agregar múltiples fuentes gráficamente y se reflejan en el sistema como puntos rojos en la malla de simulación. La malla de simulación es el cuadro localizado en la parte de abajo y es totalmente configurable.

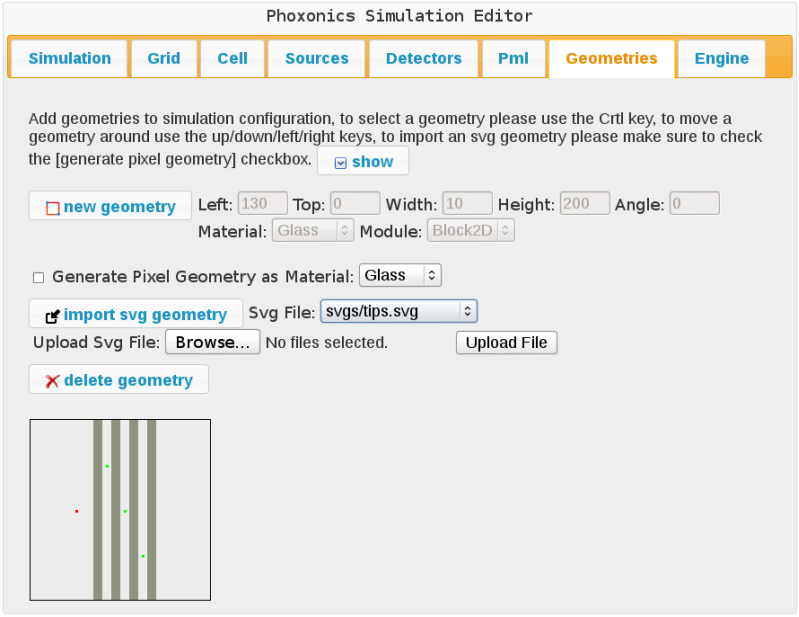

Figura 4.18: Frontend phoxonics - geometries

En la figura 4.18 se muestran las geometrías. En el cuadro de abajo se muestra que ya se han agregado cuatro geometrías de vidrio. A continuación se muestran los resultados de la simulación hecha por medio del front-end

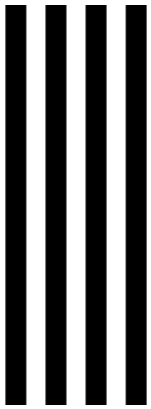

Figura 4.19: Simulación - geometrías

En la figura 4.19 se muestra la geometría de la simulación generada a partir de los archivos de datos HDF5. En este caso se generaron cuatro cuadros de vidrio y una fuente que está emitiendo campo.

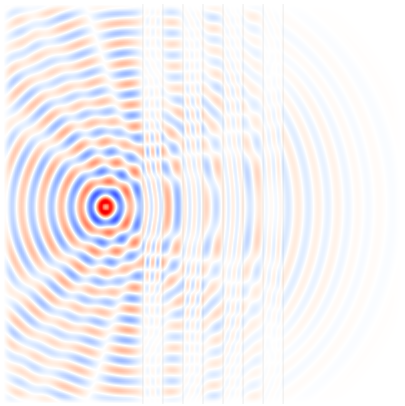

Figura 4.20: Simulación - campo electromagnético

En la figura 4.20 se muestra un corte de la simulación en el tiempo en donde se puede ver el campo electromagnético generado por una fuente sinusoidal en un medio de vacío interactuando con cuatro geometrías de vidrio

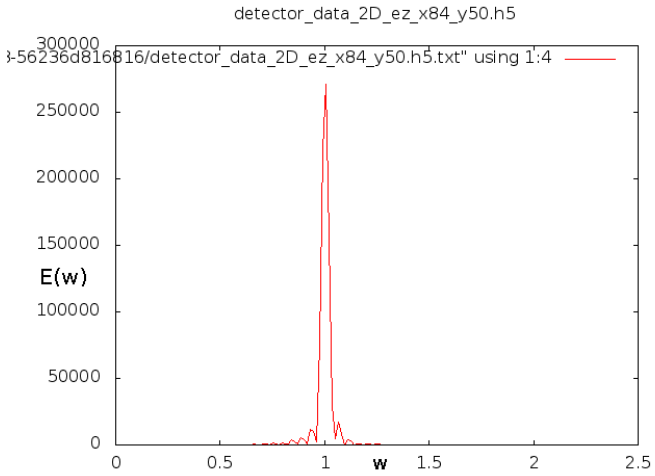

Figura 4.21: Simulación - datos de detector

Y por último en la figura 4.21 se muestra la gráfica del detector generada mediante los datos y la aplicación de la transformada de Fourier.

4.7. Conclusiones

Se ha llegado a la conclusión de que no hay mejor software a la medida que el que se hace en casa con las necesidades que se tengan a la mano. Cuando se tiene clara la funcionalidad que se quiere de un sistema, vale la pena invertir tiempo y recursos para lograr el objetivo del diseño. La idea de implementar una arquitectura de un sistema totalmente desacoplada, usando buenas

prácticas, patrones de diseño y teniendo en cuenta una analogía similar a la de los bloques legos, ha probado ser, por mucho, lo que se esperaba.

Es interesante ver la manera en que la parte gráfica administra las geometrías en la malla de simulación y la funcionalidad de poder importar geometrías en formato SVG. Teniendo en cuenta que la parte gráfica puede ser intercambiable ya que las librerías de Phoxonics también están desacopladas de esta parte.

Se tiene que tener en cuenta que el mantenimiento de este proyecto puede crecer exponencialmente ya que es un proyecto grande que puede soportar un numero ilimitado de componentes.

4.8. Referencias

[1] Umran S. Inan and Robert A. Marshall. *Numerical Electromagnetics: The FDTD Method.* Cambridge University Press, 2011.

[2] John B. Schneider. Understanding the finite-difference time-domain method. `http://www.eecs.wsu.edu/~schneidj/ufdtd`, 2010. [En Línea - Accesado: 24-Mar-2015].

[3] Phoxonics Research Group. Phoxonics (fdtd software). `http://phoxonics.com/software/`, 2015. [En Línea - Accesado: 28-Ago-2015].

[4] Martin Fowler. *Patterns of Enterprise Application Architecture.* Addison-Wesley Professional, 2002.

[5] Bjarne Stroustrup. *The C++ Programming Language, 4th Edition.* Addison-Wesley Professional, 2013.

[6] RaphaÃl Hertzog and Roland Mas. *The Debian Administrator's Handbook, Debian Wheezy from Discovery to Mastery.* Freexian SARL, 2014.

[7] David Flanagan. *JavaScript: The Definitive Guide.* O'Reilly Media, 2001.

[8] Mike O'Docherty. *Object-Oriented Analysis and Design: Understanding System Development with UML 2.0.* Wiley, 2005.

[9] Paul M. Duvall, Steve Matyas, and Andrew Glover. *Continuous Integration: Improving Software Quality and Reducing Risk.* Addison-Wesley Professional, 2007.

5 Conversión de Modos Causado por Doblamiento de Guías de Onda Fotónicas

En este capítulo se estudia la propagación de la luz en una guía de ondas en la escala de la longitud de onda donde existe un doblez. Se observa la conversión de modos de una onda guiada con un perfil simétrico que incide en el doblamiento de una guía de onda. La onda guiada que sale del doblez es una mezcla de modos simétricos y antisimétricos. La conversión de modos a través del doblamiento se cuantifica mediante el cálculo de la transformada de Fourier del perfil del campo eléctrico. Se descubre que la tasa de conversión del doblamiento es una función del ángulo.

5.1. Introducción

Las Fibras Ópticas (FO) son guías de onda dieléctricas con forma de cilindros con un diámetro (d) más grande que la longitud de onda en el rango del visible ($\lambda = 390nm$ a $\lambda = 700nm$) [1]. Las FO son flexibles, sin embargo el ángulo de doblamiento (θ) está limitado por un ángulo crítico (θ_c) definido por el principio de Reflexión Total Interna (RTI) [1]. Las FO son ampliamente utilizadas en redes ópticas, sensores y otros dispositivos [2, 3]. Hoy en día para controlar el flujo de datos en la industria de las telecomunicaciones dispositivos ópticos y electrónicos coexisten. Por un lado, para distancias largas los fotones son transportados a través de fibras ópticas que soportan altos anchos de banda con bajas pérdidas. Por otro lado, en circuitos electrónicos, las funciones esenciales de procesamiento de información son hechas mediante la manipulación del flujo de electrones. El cuello de botella para la transferencia de datos en esta tecnología híbrida es la frecuencia de operación de los procesadores electrónicos que en la actualidad es de unos pocos GHz [4].

Para aumentar la velocidad de transferencia de datos, se ha propuesto el desarrollo de Circuitos Fotónicos Integrados (CFI) [5, 6]. Un CFI hace las mismas funciones que los Circuitos Electrónicos Integrados (CEI). La diferencia es que los CFI proporcionan una manipulación directa de los fotones que viajan a través de guías de onda ópticas. Se espera que el procesamiento de la luz en CFI será capaz de superar las limitaciones físicas de la velocidad y la disipación de potencia que

enfrentan los CEI [7].

En las últimas dos décadas se ha hecho un desarrollo progresivo de los CFI que se puede clasificar en tres enfoques principales. Un primer enfoque se basa en el uso de Cristales Fotónicos (CF) como base para el diseño de dispositivos fotónicos para controlar el flujo de luz [6]. Los CF son estructuras dieléctricas periódicas compuestas por al menos dos materiales en la celda unitaria de periodo (a) que está en el orden de la longitud de onda del visible ($a \sim \lambda$) [8]. La característica fundamental de los CF es la existencia de Bandas Prohibidas Fotónicas (BPF), donde los fotones no pueden propagarse a través de la red periódica. Dispositivos fotónicos del orden de la longitud de onda del visible pueden ser diseñados en los CF introduciendo estados localizados en las BPF a través de la incorporación de defectos [9].

Una segunda estrategia para diseñar CFI se basa en el uso de plasmones [10]. Los enfoques basados en plasmones requieren una manipulación precisa de los campos electromagnéticos, que decaen exponencialmente desde las superficies [11]. Se ha sugerido la posibilidad de crear circuitos de plasmones de superficie con componentes fotónicos más pequeños que el límite de difracción de la luz [12]. Sin embargo, los plasmones tienen pérdidas ópticas sustanciales que pueden limitar seriamente su aplicabilidad [13].

Para superar estas pérdidas, se ha sugerido recientemente el uso de estructuras de ganancia-pérdida que introducen posibilidades inusuales para la propagación de plasmones [14].

Un tercer enfoque para fabricar CFI se basa en el uso de nanoalambres [15, 16, 17]. De manera similar a las FO, los nanoalambres dieléctricos son guías de onda cilíndricas de diámetro d. Sin embargo, las FO tienen un diámetro mucho mayor que la longitud de onda del visible ($d > \lambda$) que permite una propagación multimodal. En contraste, el diámetro de los nanoalambres es menor que la longitud de onda del visible ($d < \lambda$) y, como consecuencia, sólo se permite la propagación de unos pocos modos. Esta característica es deseable para la fabricación de CFI debido a que es posible tener un control y manipulación más preciso de la luz.

En la última década se ha hecho un esfuerzo intenso para reducir el diámetro de las guías de onda [17]. La fabricación de guías de onda ópticas con diámetros más pequeños que la longitud de onda es una tarea experimental difícil. Por muchos años no había sido posible lograr fabricar nanoalambres de buena calidad debido al desorden inherente en el proceso de fabricación tal como la existencia de rugosidad en la superficie y variaciones de diámetro [16]. En el año 2003, L. Tong *et al.* reportaron una técnica experimental que permitió fabricar alambres con diámetros tan pequeños como $50nm$ con una calidad excelente [17]. Estos nanoalambres permiten un mayor confinamiento de fotones en regiones muy pequeñas abriendo nuevas posibilidades tanto para el estudio de las propiedades fundamentales así como su integración en dispositivos funcionales [17, 18]. El uso de guías de onda en la escala del nanómetro ha hecho dar un giro al desarrollo de al menos cinco bloques de construcción fundamentales para circuitos fotónicos; i) fuentes de luz (láser) [19], ii) componentes pasivos (interconexiones) [20], iii) componentes activos (transistores) [21], iv) aisladores ópticos no recíprocos (diodos) [22], y v) detectores de luz [23].

Recientemente T. Voss *et al.* investigaron la union de nanoalambres que eran diferentes en

materiales y diámetros [24]. Uno de los principales resultados fue la evidencia experimental de la excitación de modos de alto orden como función de la alineación entre nanoalambres. Otro resultado importante fue la observación de la generación de modos de alto orden en las imperfecciones de un nanoalambre. Estos resultados confirmaron experimentalmente los fenómenos de la conversión de modos en nanoalambres.

La conversión de modos ha sido estudiada recientemente para otros dos casos de doblamiento en guías de onda. En el primer caso, X. Xing *et al.* analizaron la propagación de ondas de spin pasando a través de una guía magnética doblada a escala submicrométrica utilizando simulaciones micromagnéticas [25]. En el segundo caso, Q. Zhang *et al.* estudiaron teóricamente y experimentalmente la conversión de modos $TE_{10} - TE_{20}$ en una guía de onda rectangular en el régimen de microondas [26]. En estos casos se reportó la posibilidad de separar los campos después de la conversión de modos en dos diferentes guías de onda. En ambos casos, se encontró que es posible tener un desplazamiento de fase entre las dos guías como resultado de la diferencia de trayectoria. En particular, en la Ref. [25] se demostró la posibilidad de crear una especie de interferómetro Mach-Zehnder donde las compuertas lógicas NOT y AND se pueden sintonizar en frecuencia. Este resultado abre la posibilidad de integrar la conversión de modos en dispositivos interferométricos.

5.2. Teoría

Se analiza la propagación de los campos electromagnéticos en una guía de onda plana utilizando las ecuaciones rotacionales de Maxwell en el sistema MKS

$$\nabla \times \mathbf{H}(\mathbf{x}, t) = \frac{\partial}{\partial t}\mathbf{D}(\mathbf{x}, t), \tag{5.1}$$

$$\nabla \times \mathbf{E}(\mathbf{x}, t) = -\frac{\partial}{\partial t}\mathbf{B}(\mathbf{x}, t). \tag{5.2}$$

Para la ecuación 5.1 se ha considerado el caso donde la densidad de corriente es cero, $\mathbf{J}(\mathbf{x}, t) = 0$. Las ecuaciones de Maxwell están complementadas por las relaciones constitutivas que introducen las propiedades electromagnéticas del material del problema físico. La permitividad eléctrica (ε) provee una descripción de la interacción macroscópica entre el campo eléctrico y el material dieléctrico mientras que la permeabilidad magnética (μ) describe la interacción del material con el campo magnético. Para el caso de un material dieléctrico (sin dispersión) estas relaciones se pueden escribir en la forma

$$\mathbf{D}(\mathbf{x}, t) = \varepsilon(\mathbf{x})\mathbf{E}(\mathbf{x}, t), \tag{5.3}$$

$$\mathbf{B}(\mathbf{x}, t) = \mu(\mathbf{x})\mathbf{H}(\mathbf{x}, t). \tag{5.4}$$

En la figura 5.1 se presenta la geometría básica de una guía de onda plana compuesta de un bloque de alto índice de refracción (n_h) de diámetro d entre dos medios semi-infinitos de bajo índice de refracción (n_l). En este trabajo se analiza el caso de la polarización Transversal Eléctrico (TE) donde el campo eléctrico es paralelo al eje z, $\mathbf{E}(\mathbf{x},t) = \hat{k}E_z(x,y,t)$. Las ondas electromagnéticas dentro y fuera del bloque se propagan y decaen respectivamente.

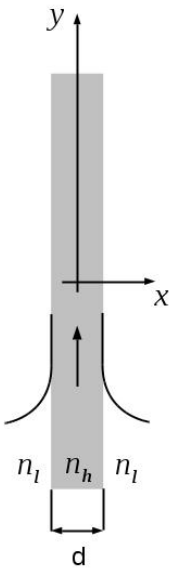

Figura 5.1: Guía de onda plana de anchura d e índice de refracción n_h entre dos medios semi-infinitos de índice de refracción n_l.

La relación de dispersión se obtiene resolviendo numéricamente dos ecuaciones trascendentes [27]. Por un lado, la condición para los eigenmodos pares con respecto al eje x está dada por la ecuación

$$\tan(\pi Q_{x,h}) = \frac{Q_{x,l}}{Q_{x,h}} \tag{5.5}$$

Por otro lado, la condición para los eigenmodos impares es

$$\cot(\pi Q_{x,h}) = -\frac{Q_{x,l}}{Q_{x,h}} \tag{5.6}$$

Los vectores de onda reducidos en el eje x para los medios de alto y bajo índice de refracción son

$$Q_{x,h} = \sqrt{n_h^2\Omega^2 - Q_y^2}, \tag{5.7}$$

$$Q_{x,l} = \sqrt{Q_y^2 - n_l^2\Omega^2}. \tag{5.8}$$

La frecuencia reducida es

$$\Omega = \frac{\omega d}{2\pi c} \tag{5.9}$$

Y el vector de onda reducido en el eje y es

$$Q_y = \frac{k_y d}{2\pi} \tag{5.10}$$

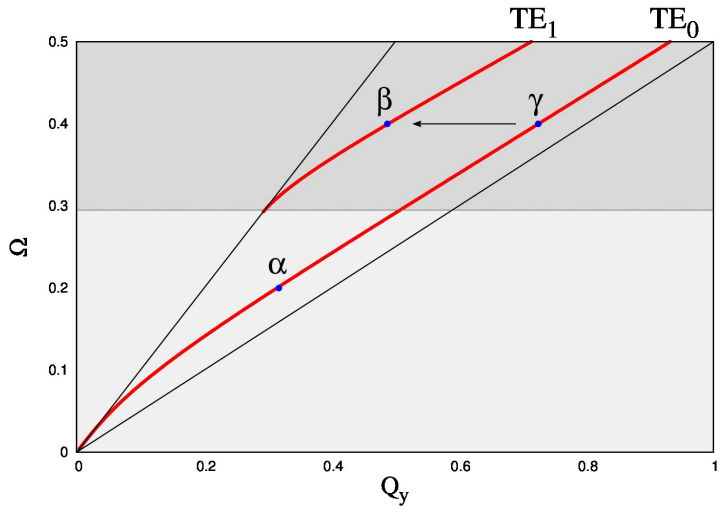

Figura 5.2: Relación de dispersión para una guía de onda plana. Las regiones gris claro y obscuro son los regímenes monomodal y multimodal. Los puntos α y γ pertenecen al modo TE_0 mientras que el punto β es parte del modo TE_1.

La Figura 5.2 presenta la relación de dispersión para la guía de onda plana. Los índices de refracción altos y bajos son $n_h = 2$ y $n_l = 1$, que corresponde al índice de refracción de nanoalambres de ZnO en aire, de acuerdo con las referencias [18, 24]. Q_y y Ω son los ejes de abscisas y ordenadas, respectivamente. La zona gris claro ($\Omega < 0.3$) es el régimen de frecuencia monomodal, donde sólo se permite un eigenmodo. La zona gris obscuro ($\Omega > 0.3$) es el régimen de frecuencia multimodal donde se permiten dos o más eigenmodos. Los modos TE_0 y TE_1 corresponden a las primeras soluciones pares e impares, respectivamente. En el régimen monomodal, se identifica el eigenmodo α en (0.32, 0.2) el cual pertenece al modo TE_0. En la región multimodal se identifica el eigenmodo β en (0.49, 0.4) y el eigenmodo γ en (0.72, 0.4) que corresponden a los modos TE_1 y TE_0, respectivamente. La flecha que va desde el punto γ hasta el punto β ilustra la conversión de modos que se analiza a continuación.

En la figura 5.3 se presenta el perfil de campo eléctrico para los eigenmodos α, β y γ. Se observa que los modos TE_0 correspondientes a los eigenmodos α y γ en la figura 5.3(a) y figura 5.3(c) tienen una simetría par. De manera correspondiente el modo TE_1 que corresponde al eigenmodo β en la figura 5.3(b) tiene una simetría impar.

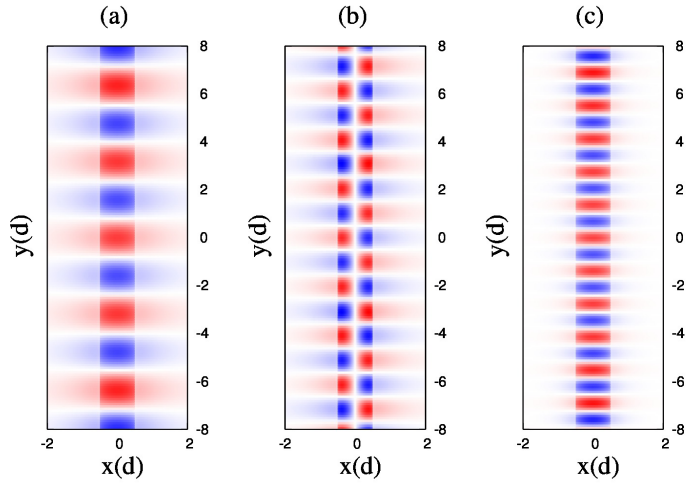

Figura 5.3: Perfil de campo eléctrico para los eigenmodos α, β y γ se presentan en (a), (b) y (c).

5.3. Método Numérico

Se simula la propagación de los campos electromagnéticos en la guía de onda plana utilizando el Método de Diferencias Finitas en el Dominio del Tiempo (FDTD) [28]. El método FDTD es una técnica computacional usada para modelar la evolución temporal de las ecuaciones de Maxwell. Este método numérico está basado en la formulación de las ecuaciones diferenciales en un sistema asociado de ecuaciones en diferencias finitas. Las ecuaciones de Maxwell dependientes del tiempo se discretizan utilizando una aproximación en diferencias centrales para las derivadas parciales de espacio y tiempo. El sistema de ecuaciones en diferencias finitas resultante es reformulado en términos de un algoritmo computacional que puede ser resuelto en el dominio del tiempo mediante un proceso recursivo.

En este trabajo, se ha simulado la propagación del campo electromagnético utilizando Phoxonics, el software que hemos diseñado para implementar la técnica FDTD [29]. Phoxonics es un programa computacional sofisticado y flexible que permite estudiar diferentes configuraciones geométricas. En particular se ha analizado la conversión de modos causada por doblamiento en una guía de onda plana.

Considerando una polarización del campo eléctrico $\mathbf{E}(\mathbf{x},t) = \hat{k}E_z(x,y,t)$, las ecuaciones de Maxwell 5.1 y 5.2 se pueden escribir como tres ecuaciones escalares en la forma

$$\frac{\partial}{\partial t}D_z(x,y,t) = \frac{\partial}{\partial x}H_y(x,y,t) - \frac{\partial}{\partial y}H_x(x,y,t), \tag{5.11}$$

$$\frac{\partial}{\partial y}E_z(x,y,t) = -\frac{\partial}{\partial t}B_x(x,y,t), \tag{5.12}$$

$$\frac{\partial}{\partial x}E_z(x,y,t) = \frac{\partial}{\partial t}B_y(x,y,t). \tag{5.13}$$

El grupo de ecuaciones escalares (5.11, 5.12, 5.13) puede ser reformulado en términos de diferencias finitas. La versión en diferencias finitas de la ecuación 5.11 en el punto $(x,y,t) = (i\Delta x, j\Delta y, n\Delta t)$ es

$$\frac{D_z(i,j,n+1/2) - D_z(i,j,n+1/2)}{\Delta t} = \frac{H_y(i+1/2,j,n) - H_y(i-1/2,j,n)}{\Delta x} - \frac{H_x(i,j+1/2,n) - H_x(i,j-1/2),n}{\Delta y} \tag{5.14}$$

La ecuación 5.12 para el punto $(x,y,t) = [i\Delta x, (j+1/2)\Delta y, (n+1/2)\Delta t]$ en diferencias finitas se escribe de la siguiente forma

$$\frac{B_x(i,j+1/2,n+1) - B_x(i,j+1/2,n)}{\Delta t} = \frac{E_z(i,j+1,n+1/2) - E_z(i,j,n+1/2)}{\Delta y} \tag{5.15}$$

Finalmente la ecuación 5.13 en diferencias finitas para el punto $(x,y,t) = [(i+1/2)\Delta x, j\Delta y, (n+1/2)\Delta t]$ es

$$\frac{B_y(i+1/2,j,n+1) - B_y(i+1/2,j,n)}{\Delta t} = \frac{E_z(i+1,j,n+1/2) - E_z(i,j,n+1/2)}{\Delta x} \tag{5.16}$$

Para establecer el grupo de ecuaciones en diferencias finitas (5.14, 5.15, 5.16) en términos de un algoritmo numérico, se escriben las ecuaciones a manera *escalonada* para poderlas resolver recursivamente mediante un programa computacional. Para validar la implementación computacional del método FDTD utilizando Phoxonics, se considera una guía de onda plana con una o dos fuentes monocromáticas embebidas. Estas fuentes son colocadas en la parte de abajo de la malla de simulación para permitir ver la excitación de los eigenmodos. Para el caso de los modos pares α y γ, los modos pueden ser excitados con una sola fuente monocromática definida por

$$E_z(x,y,t) = A_1\delta(x-w_{1x})\delta(y-w_{1y})sin(\omega t) \tag{5.17}$$

Para excitar el modo β, es necesario considerar dos fuentes monocromáticas definidas por

$$E_z(x,y,t) = [A_1\delta(x-w_{1x})\delta(y-w_{1y}) + A_2\delta(x-w_{2x})\delta(y-w_{2y})]sin(\omega t) \tag{5.18}$$

Los puntos (w_{1x}, w_{1y}) y (w_{2x}, w_{2y}) definen la posición de las fuentes monocromáticas en los ejes x y y. A_1 y A_2 son amplitudes de las fuentes. La frecuencia de las fuentes monocromáticas es ω.

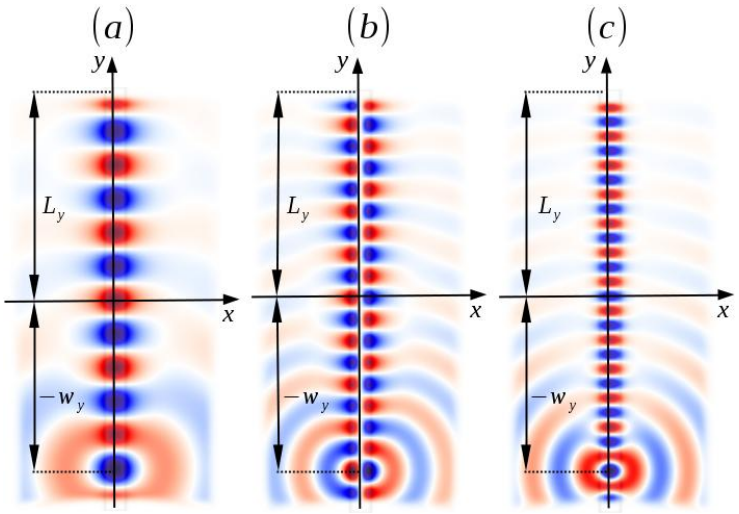

Figura 5.4: Simulación de la propagación del campo eléctrico mediante FDTD. La excitación de los eigenmodos α, β y γ se presentan en (a), (b) y (c), respectivamente.

La excitación de los eigenmodos α y γ mediante FDTD se obtiene considerando una sola fuente monocromática, como está definido en la ecuación 5.17. La posición de la fuente monocromática es $(w_{1x}, w_{1y}) = (0, -8d)$. En la figura 5.4(a) y 5.4(c) la distribución espacial del modo TE_0 se ilustra cuando las fuentes están emitiendo en las frecuencias reducidas $\Omega = 0.2$ y $\Omega = 0.4$, respectivamente. Para obtener la excitación del eigenmodo β se necesitan usar dos fuentes monocromáticas como está definido en la ecuación 5.18. Las amplitudes de las fuentes son $A_1 = 1$ y $A_2 = -1$. Las posiciones de las fuentes son $(w_{1x}, w_{1y}) = (-0.4d, -8d)$ y $(w_{2x}, w_{2y}) = (+0.4d, -8d)$, respectivamente. En la figura 5.4(b) se presenta la distribución del modo TE_1 para el caso en donde ambas fuentes monocromáticas están emitiendo en una frecuencia reducida de $\Omega = 0.4$

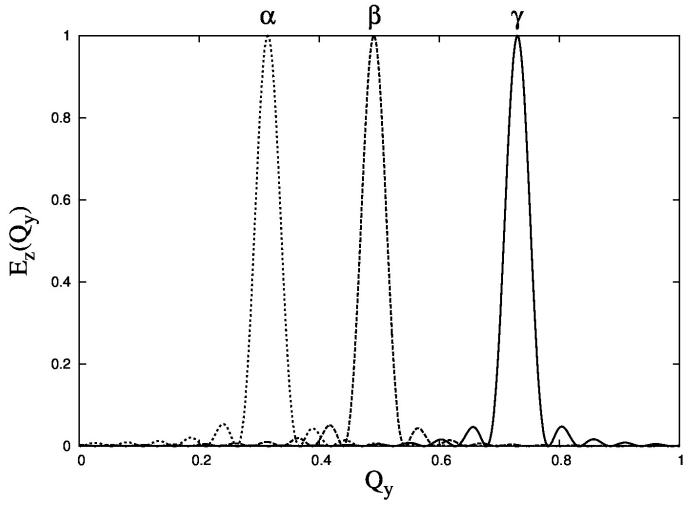

Figura 5.5: Transformada de Fourier del campo eléctrico en la superficie de la guía de onda plana. De izquierda a derecha las lineas punteadas, cortadas y sólidas son los modos α, β y γ.

Para validar la excitación de los modos α, β y γ en la guía de onda plana usando la técnica del FDTD, se calcula la Transformada de Fourier (TF) del campo eléctrico usando la fórmula

$$E_z(Q_y) \cong \frac{1}{\sqrt{2\pi}} \int_0^{L_y} E_z(y) e^{i2\pi Q_y y/d} dy \tag{5.19}$$

Los límites de integración han sido cambiados para considerar el intervalo finito $[0 : L_y]$. En la figura 5.5 se presenta la TF para el campo eléctrico en la superficie de la guía de onda plana. La TF de la distribución de los campos eléctricos en la figura 5.3(a), 5.3(b) y 5.3(c) se presenta en la figura 5.5 con líneas punteadas, cortadas y sólidas, respectivamente. De izquierda a derecha, los picos de la TF alrededor de los vectores de onda reducidos $Q_y = 0.32$, $Q_y = 0.49$ y $Q_y = 0.72$ prueban la existencia de los eigenmodos α, β y γ, respectivamente.

5.4. Conversión de Modos

En esta sección se analiza la conversión de modos causado por doblamiento en una guía de onda plana. Se considera el caso donde el modo TE_0 incide en el doblez. En la figura 5.6 se muestra una guía de onda plana con un ángulo de doblez θ cercano al origen del sistema de coordenadas.

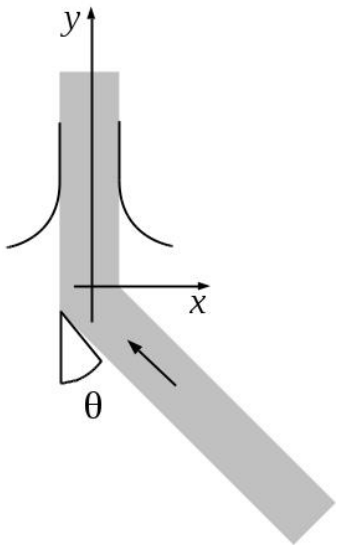

Figura 5.6: Guía de onda plana con un ángulo de doblez θ. El doblez se coloca en la cercanía del origen del sistema de coordenadas.

En la Figura 5.7 se simula la propagación de la luz en el régimen monomodal. Un perfil de haz par correspondiente al modo TE_0 con una frecuencia reducida de $\Omega = 0.2$ está incidiendo en el doblez de la guía de onda. Los casos de guías de onda con doblez de ángulos de $\theta = 15^0$, $\theta = 30^0$, y $\theta = 90^0$ se presentan en (a), (b) y (c), respectivamente. Hay tres observaciones importantes. La primera es que la onda guiada incidente pasa a través del doblez sin distorsión en su simetría.

La segunda es que la propagación de la luz guiada no está limitada por ningún ángulo de doblez crítico como existe en la FO convencional como consecuencia de la RTI. Y la tercera es, que la emisión en el doblez se incrementa con el ángulo de doblez.

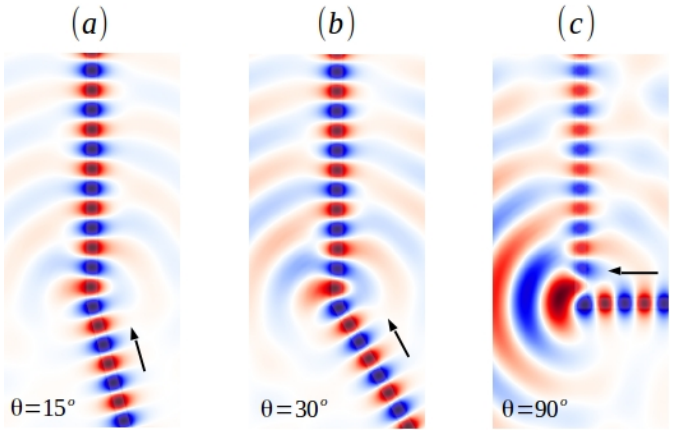

Figura 5.7: Propagación en el régimen monomodal ($\Omega = 0.2$) del modo TE_0 a través de una guía de onda con un doblez angular θ. Los casos $\theta = 15^0$, $\theta = 30^0$, y $\theta = 90^0$ se presentan en (a), (b) y (c).

En la Figura 5.8 se simula la propagación en el régimen multimodal. Aquí el modo TE_0 está incidiendo en el doblez con una frecuencia reducida de $\Omega = 0.4$. Para el caso de $\theta = 15^0$ ilustrado en la figura 5.8(a), se observa que existe una ligera deformación en la geometría del perfil del campo eléctrico. En la figura 5.8(b) se considera el caso para $\theta = 30^0$, para el que se observa que después del doblez hay un perfil mixto de modos TE_0 y TE_1, donde no es posible identificar una simetría par o impar predominante. Por último, en la figura 5.8(c) se presenta el caso para $\theta = 90^0$. Es evidente que el modo entrante TE_0 es convertido después del doblez en el modo TE_1.

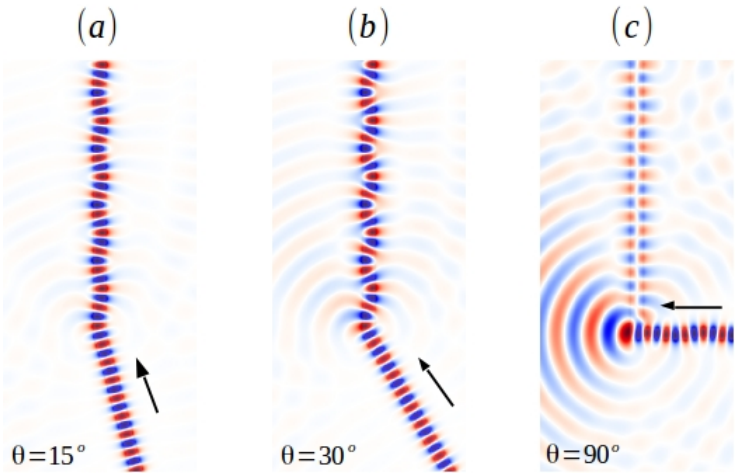

Figura 5.8: Propagación en el régimen multimodal ($\Omega = 0.4$) del modo TE_0 a través de una guía de ondas con un doblez angular θ. Los casos $\theta = 15^0$, $\theta = 30^0$ y $\theta = 90^0$ se muestran en (a), (b) y (c).

La conversión de modos se puede cuantificar mediante la Transformada de Fourier (TF) del

campo eléctrico como está definido en la ecuación 5.19. La TF se calcula para el campo eléctrico después del doblez en la superficie de la guía de onda plana. Se considera el caso presentado en la figura 5.8 donde una onda guiada del modo TE_0 está incidiendo en el doblez con una frecuencia reducida de $\Omega = 0.4$

La Figura 5.9 presenta la TF para los casos correspondientes a un ángulo de doblez de $\theta = 0^0$, $\theta = 15^0$, $\theta = 30^0$, y $\theta = 60^0$. Usando líneas negras, azules, verdes y rojas, respectivamente. Se observa que a medida que aumenta el ángulo de doblez, la amplitud que corresponde al modo γ decrece. A la inversa, de manera que el ángulo de doblez aumenta la amplitud del modo β aumenta. Por lo tanto después del doblez, la amplitud del modo TE_0 decrece y el modo TE_1 se incrementa, en ambos casos como función del ángulo de doblez.

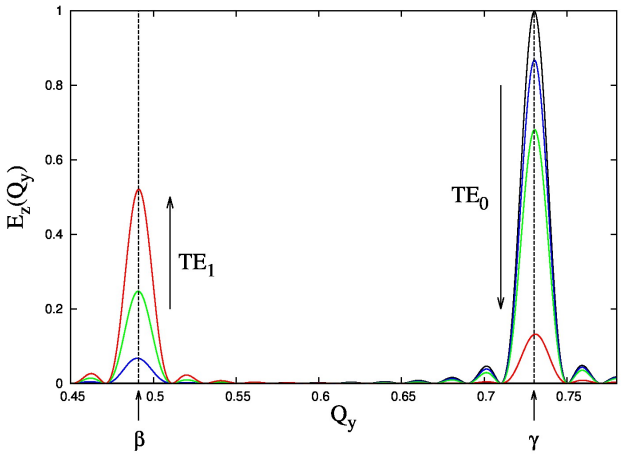

Figura 5.9: TF del perfil del campo eléctrico después del doblez. Los casos $\theta = 0^0$, $\theta = 15^0$, $\theta = 30^0$ y $\theta = 90^0$ se presentan con líneas negras, azul, verde y rojo, respectivamente.

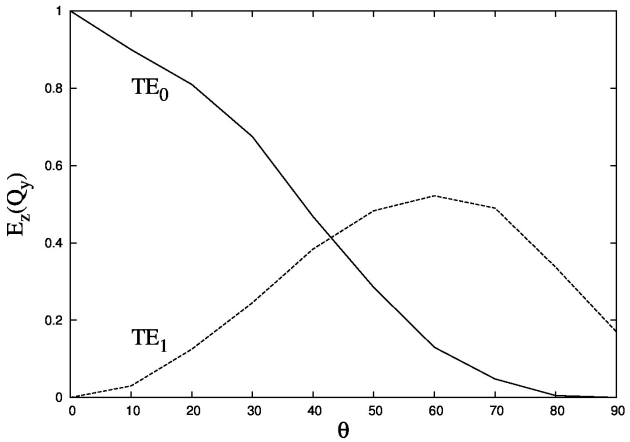

Figura 5.10: Transformada de Fourier del perfil del campo eléctrico como función de θ. La amplitud de los modos TE_0 y TE_1 se presentan con líneas sólidas y punteadas, respectivamente.

En la figura 5.10 se presenta la variación de los modos TE_0 y TE_1 como una función del ángulo θ utilizando líneas sólidas y punteadas, respectivamente. Se observa que la cantidad de modo TE_0 disminuye monótonamente como una función del ángulo de doblez. Se descubre que el modo TE_0 desaparece después del doblez cuando se alcanza $\theta = 80^0$. En contraste, la cantidad de modo TE_1 aumenta como una función del ángulo de doblez en el rango desde 0^0 a 60^0. A 60^0 la cantidad de modo TE_1 empieza a disminuir. Se observa que para el ángulo de doblez $\theta = 90^0$ se produce una conversión de modo completa.

5.5. Conclusiones

En conclusión, estos resultados demuestran que las guías de onda nanométricas con doblez no solo transmiten luz pasivamente desde un punto a otro. En el régimen multimodal, el doblez de una guía de onda nanométrica actúa como un convertidor de modos excitando modos de alto orden. Por lo tanto, el doblez de la guía de onda debe ser considerado intrínsecamente como un dispositivo activo debido a que provoca una redistribución espacial de los campos. Este fenómeno ofrece una oportunidad de crear dispositivos lógicos basados en la conversión de modos.

Se ha descubierto una estrategia para identificar la conversión de modos mediante el cálculo de la transformada de Fourier de la distribución del campo eléctrico después del doblez. Finalmente, este estudio basado en el análisis de una geometría de dos dimensiones es un primer paso para entender mejor y utilizar la conversión de modos en nanoalambres tridimensionales, el cual es un paso esencial para el diseño de CFI basado en nanoalambres.

5.6. Referencias

[1] John David Jackson. *Classical Electrodynamics Third Edition*. Wiley, 1998.

[2] J.P Goure and I Verrier. *Optical Fibre Devices (Series in Optics and Optoelectronics)*. CRC Press, 2001.

[3] Limin Tong, Fei Zi, Xin Guo, and Jingyi Lou. Optical microfibers and nanofibers: A tutorial. *Optics Communications*, 285(23):4641 – 4647, 2012. Special Issue: Optical micro/nanofibers: Challenges and Opportunities.

[4] James D. Meindl. Beyond moore's law: The interconnect era. *Computing in Science and Engg.*, 5(1):20–24, January 2003.

[5] Larry A. Coldren, Scott W. Corzine, and Milan L. Mashanovitch. *Diode Lasers and Photonic Integrated Circuits*. Wiley, 2012.

[6] J. D. Joannopoulos, Pierre R. Villeneuve, and Shanhui Fan. Photonic crystals: putting a new twist on light. *Nature*, 386(6621):143–149, mar 1997.

[7] Xin Guo, Yibin Ying, and Limin Tong. Photonic nanowires: From subwavelength waveguides to optical sensors. *Acc. Chem. Res.*, 47(2):656–666, feb 2014.

[8] Eli Yablonovitch. Inhibited spontaneous emission in solid-state physics and electronics. *Phys. Rev. Lett.*, 58:2059–2062, May 1987.

[9] John D. Joannopoulos, Steven G. Johnson, Joshua N. Winn, and Robert D. Meade. *Photonic Crystals: Molding the Flow of Light*. Princeton University Press, 2008.

[10] Shinji Hayashi and Takayuki Okamoto. Plasmonics: visit the past to know the future. *Journal of Physics D: Applied Physics*, 45(43):433001, oct 2012.

[11] E. Ozbay. Plasmonics: Merging photonics and electronics at nanoscale dimensions. *Science*, 311(5758):189–193, jan 2006.

[12] Thomas W. Ebbesen, Cyriaque Genet, and Sergey I. Bozhevolnyi. Surface-plasmon circuitry. *Physics Today*, 61(5):44–50, may 2008.

[13] M. Kuttge, E. J. R. Vesseur, J. Verhoeven, H. J. Lezec, H. A. Atwater, and A. Polman. Loss mechanisms of surface plasmon polaritons on gold probed by cathodoluminescence imaging spectroscopy. *Applied Physics Letters*, 93(11):113110, 2008.

[14] Jesus Manzanares-Martinez, Carlos Ivan Ham-Rodriguez, Damian Moctezuma-Enriquez, and Betsabe Manzanares-Martinez. Omnidirectional mirror based on bragg stacks with a periodic gain-loss modulation. *AIP Advances*, 4(1):017136, jan 2014.

[15] Peter J. Pauzauskie and Peidong Yang. Nanowire photonics. *Materials Today*, 9(10):36 – 45, 2006.

[16] F. Ladouceur. Roughness, inhomogeneity, and integrated optics. *J. Lightwave Technol.*, 15(6):1020–1025, jun 1997.

[17] Limin Tong, Rafael R. Gattass, Jonathan B. Ashcom, Sailing He, Jingyi Lou, Mengyan Shen, Iva Maxwell, and Eric Mazur. Subwavelength-diameter silica wires for low-loss optical wave guiding. *Nature*, 426(6968):816–819, dec 2003.

[18] Tobias Voss, Geoffry T. Svacha, Eric Mazur, Sven Müller, Carsten Ronning, Denan Konjhodzic, and Frank Marlow. High-order waveguide modes in ZnO nanowires. *Nano Letters*, 7(12):3675–3680, dec 2007.

[19] Deirdre O'Carroll, Ingo Lieberwirth, and Gareth Redmond. Microcavity effects and optically pumped lasing in single conjugated polymer nanowires. *Nature Nanotech*, 2(3):180–184, feb 2007.

[20] M. Law. Nanoribbon waveguides for subwavelength photonics integration. *Science*, 305(5688):1269–1273, aug 2004.

[21] Devin Powell. Light flips transistor switch. *Nature*, 498(7453):149–149, jun 2013.

[22] Dirk Jalas, Alexander Petrov, Manfred Eich, Wolfgang Freude, Shanhui Fan, Zongfu Yu, Roel Baets, Miloš Popović, Andrea Melloni, John D. Joannopoulos, Mathias Vanwolleghem, Christopher R. Doerr, and Hagen Renner. What is — and what is not — an optical isolator. *Nature Photonics*, 7(8):579–582, jul 2013.

[23] J. Wang. Highly polarized photoluminescence and photodetection from single indium phosphide nanowires. *Science*, 293(5534):1455–1457, aug 2001.

[24] Margit Zacharias, Hong Jin Fan, and Rolf Haug. *Advances in Solid State Physics 48*. Springer, 2008.

[25] Xiangjun Xing, Yongli Yu, Shuwei Li, and Xiaohong Huang. How do spin waves pass through a bend? *Scientific Reports*, 3, oct 2013.

[26] Qiang Zhang, Cheng-Wei Yuan, and Lie Liu. Theoretical design and analysis for rectangular waveguide mode converters. *IEEE Transactions on Microwave Theory and Techniques*, 60(4):1018–1026, apr 2012.

[27] Pochi Yeh. *Optical Waves in Layered Media*. Wiley-Interscience, 2nd edition, 3 2005.

[28] Kane S Yee et al. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Trans. Antennas Propag*, 14(3):302–307, 1966.

[29] Phoxonics Research Group. Phoxonics (fdtd software). `http://phoxonics.com/software/`, 2015. [En Línea - Accesado: 28-Ago-2015].

6 | Conclusiones, Resultados y Trabajo Futuro

6.1. Remembranza

- En esta tesis se han presentado los conceptos básicos sobre nanotecnología, nanotecnología computacional, estructuras nanométricas y nanofotónica computacional entre otros. Se establecen las razones por las cuales es importante la investigación de estructuras a nanoescala. Se define la manera en la cual se pueden perfeccionar las herramientas computacionales para resolver sistemas de investigación relacionados con la nanotecnología computacional.
- Se ha mostrado la forma analítica de resolver sistemas físicos. Se da a conocer el álgebra necesaria para describir los fenómenos que se quieren estudiar tomando como herramientas la física y la matemática. Se establece la base teórica desde la cual se pueden construir análisis numéricos y que sirve para comprobar la eficacia de los mismos. Se establece la plataforma teórica-matemática que sirve para analizar y estudiar los distintos sistemas propuestos.
- Se ha descrito el método FDTD y cómo sus avances promueven inmensamente el desarrollo del campo de el electromagnetismo computacional y cómo éste juega un papel cada vez más fundamental en las aplicaciones de la nanotecnología. Se establece que el método FDTD es una herramienta indispensable para modelar diversos sistemas y materiales. Se sugiere que el método FDTD será ampliamente adoptado para simular múltiples fenómenos físicos, a través del cual las propiedades ópticas, eléctricas, térmicas, mecánicas y cuánticas de componentes y dispositivos puedan ser investigadas.

6.2. Resultados

Los resultados de este trabajo se presentan a continuación

- Uno de los resultados de esta tesis es la escritura y publicación de un libro titulado *Simulación Computacional de Nanoestructuras con Meep: Técnicas, análisis y modelado computacional en Meep con el método (FDTD)* [1]. En este libro se refleja un esfuerzo hecho en el área de

investigación de mecanismos implementados por un software del MIT llamado MEEP que utiliza la técnica FDTD.

- Otro de los resultados es la creación de Phoxonics (http://phoxonics.com/software/). Phoxonics es un software computacional sofisticado y flexible que utiliza el método FDTD para resolver cálculos electromagnéticos. Es creado a partir de la necesidad de analizar fácilmente estructuras geométricas personalizadas y la visión de obtener la posibilidad de intercambiar motores de cálculo para resolver múltiples tipos de ecuaciones, de ahí su nombre Phoxonics (photonics y phononics) aunque la parte de phononics aun no ha sido terminada. Phoxonics implementa estas características de forma flexible ya que ha sido diseñado precisamente con esto en mente desde el inicio.
- Finalmente los resultados más importantes. Consisten en la publicación de artículos en revistas científicas que no habrían sido posibles de no haber logrado los puntos antes mencionados.

6.3. Conclusiones

Las conclusiones de esta tesis se presentan a continuación

- El método FDTD implementado en un software es una forma poderosa de simulación computacional que permite resolver las ecuaciones de Maxwell y predecir los campos electromagnéticos en estructuras a nivel nanómetro.
- El desarrollo de una herramienta computacional que permita diseñar fácilmente estructuras a nivel nanómetro no es una tarea trivial. El tiempo invertido en la creación de esta herramienta se compensa mientras sea utilizado en el estudio de fenómenos de propagación de ondas electromagnéticas.
- La conversión de modos causada por doblamiento de guías de onda fotónicas demuestra que las guías de onda nanométricas con doblez no solo transmiten luz pasivamente desde un punto a otro. En el régimen multimodal, el doblez de una guía de onda nanométrica actúa como un convertidor de modos excitando modos de alto orden. Por lo tanto, el doblez de la guía de onda debe ser considerado intrínsecamente como un dispositivo activo debido a que provoca una redistribución espacial de los campos.
- El fenómeno de la conversión de modos ofrece una oportunidad de crear compuertas lógicas. Se ha descubierto una estrategia para identificar la conversión de modos mediante el cálculo de la transformada de Fourier de la distribución del campo eléctrico después del doblez. Finalmente, este estudio basado en el análisis de una geometría de dos dimensiones es un primer paso para entender mejor y utilizar la conversión de modo en nanoalambres tridimensionales, el cual es un paso esencial para el diseño de Circuitos Fotónicos Integrados basado en nanoalambres.

6.4. Trabajo Futuro

La creación de un software implica el mantenimiento y mejoramiento perpetuo del mismo. El software creado en esta tesis puede ser mejorado y actualizado de muchas formas. En esta sección se presentan algunas de las formas más relevantes que pueden ser investigadas a partir del trabajo actual.

- **Paralelización con CPUs:** El proceso de paralelización de software con Central Processing Units (CPUs) se refiere a modificar el código del sistema de una manera en la que se puedan utilizar todos los cpu's disponibles, existen varias formas de lograr este objetivo, una de ellas es mediante Message Passing Interface (MPI) y otra forma más eficiente de manejarlo es mediante la librería Threading Building Blocks (TBB). TBB por sus siglas en inglés, ofrece un enfoque rico y completo para expresar el paralelismo en un programa de C++ [2].
- **Paralelización con GPUs:** Los GPUs han irrumpido en la escena de la computación científica como una tecnología innovadora que ha demostrado un rendimiento sustancial y una mejora en la eficiencia energética para numerosas aplicaciones científicas. Eventualmente esta tecnología será implementada en los algoritmos de cómputo científico para acelerar cálculos exponencialmente [3].
- **Diseño de Bloques Para FDTD Acústico:** Phoxonics brinda la flexibilidad de soportar bloques con lógica independiente siempre y cuando se cumpla con los contratos definidos. De esta manera se abren las puertas para crear un FDTD totalmente distinto y aventurarse a discretizar ecuaciones alternativas.
- **Análisis de Submallas:** Existen múltiples sistemas que requieren un análisis más detallado en partes específicas del sistema, esto lleva a la definición de submallas. Las submallas requieren de un tratamiento especial y algoritmos adecuados.
- **Soporte para 3 Dimensiones:** Se tiene contemplado agregar soporte a Phoxonics para poder generar simulaciones en tres dimensiones.

6.5. Referencias

[1] Yohan Rodríguez and Jesús Manzanares. *Simulación Computacional de Nanoestructuras con Meep: Técnicas, análisis y modelado computacional en Meep con el método (FDTD) (Spanish Edition).* Editorial Académica Española, 2013.

[2] James Reinders. *Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism.* O'Reilly Media, 2007.

[3] Rob Farber. *CUDA Application Design and Development.* Morgan Kaufmann, 2011.

A | Programas de Cómputo

A.1. Propagación Temporal de la Solución Armónica

$$E_z(y,t) = A_1 \cos(k_y y - \omega t) + A_2 \cos(-k_y y - \omega t) \tag{A.1}$$

Listado A.1: Ecuación de onda en una dimensión

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#include <algorithm>
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

int main() {
    int iy, Ny;
    double pi, c, l0, t, tao, w0, ky, Ep, Em;
    double y, yi, yf, dy;

    pi = 2.0 * asin(1.0);
    c = 3e8;
    l0 = 600.0e-9;
    ky = (2.0 * pi) / l0;
    tao = l0 / c;
    w0 = (2.0 * pi) / tao;

    t = 0.0 * tao;

    Ny = 100;
    yi = -2.0 * l0;
    yf = +2.0 * l0;
    dy = (yf - yi) / Ny;

    ofstream myfile;
    myfile.open("data_00.dat");

    for (iy = 0; iy <= Ny; iy++) {
        y = yi + iy * dy;
```

```
37          Ep = cos(+1.0 * ky * y);
38          Em = cos(-1.0 * ky * y);
39          myfile << y / l0 << " " << Ep << " " << Em << '\n';
40      }
41      myfile.close();
42  }
```

Listado A.2: Ecuación de onda en una dimensión (código gnuplot)

```
1 set term postscript eps enhanced color
2 set output 'data_00.eps'
3 set nokey
4
5 set grid
6 set xlabel "z ({/Symbol l}_0)" font "Times,25"
7 set ylabel "E_x(z)" font "Times,25"
8
9 plot 'data_00.dat' u 1:2 w l,'' u 1:3 w lp lc 3
```

A.2. La Línea de Luz

Listado A.3: Graficando línea de luz

```
1 set term postscript eps enhanced color
2 set output 'luz.eps'
3
4 unset key
5 set xlabel "Q_y" font "Times,15"
6 set ylabel "{/Symbol W}" font "Times,15"
7 set yra[0:1]
8 set xra[0:1]
9 set label "Soluciones oscilantes" at 0.1,0.4 font "Times,20"
10 set label "Soluciones decadentes" at 0.35,0.3 font "Times,20"
11 plot "luz.dat" u 1:2 w l lt 1 lc 7 lw 2
```

A.3. La Condición de Modos

Considere que la ecuación 2.100 que define la existencia de modos evanescentes en el medio ε_a es

$$\mathrm{Im}(Q_{ax}) \tag{A.2}$$

Esta relación define una relación entre Q_y y Ω de la forma

$$Q_y \geq n_a \Omega \tag{A.3}$$

Esta desigualdad define la condición límite

$$\Omega^{(a)} = \frac{1}{n_a} Q_y \tag{A.4}$$

De la misma forma, la existencia de soluciones armónicas en el medio ε_b requiere que se cumpla

$$\mathrm{Im}(Q_{bx}) \tag{A.5}$$

Esta relación define una relación entre Q_y y Ω de la forma

$$Q_y \leq n_b \Omega \tag{A.6}$$

Esta desigualdad define la relación

$$\Omega^{(b)} = \frac{1}{n_b} Q_y \tag{A.7}$$

El programa "lineas.cpp" genera el archivo "lineas.dat" que define las lineas $\Omega^{(a)}$ y $\Omega^{(b)}$. Usando el programa "lineas.plt" se crea la Figura A.1 que permite visualizar regiones en donde existen modos oscilantes, modos decadentes y modos guiados. En el panel (a) se presenta a la izquierda de la línea roja la región en donde los campos oscilan en el medio ε_a. En el panel (a) se presenta a la derecha de la línea azul la región en donde los campos decaen en el medio ε_b. En el panel (c) se ilustra la región entre las líneas azul y roja en la cual existen modos guiados.

Listado A.4: Líneas de luz

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>

using namespace std;

main() {
    int iy;
    float na, nb, Oma, Omb, Qy;

    na = 2;
    nb = 1;

    ofstream myfile;
    myfile.open("lineas.dat");
    for(iy = 0; iy <= 1; iy++) {
        Qy = iy * 2.0;
        Oma = (1 / na) * Qy;
        Omb = (1/nb) * Qy;
        myfile << Qy << " " << Oma << "  " << Omb << "\n";
    }
    myfile.close();
    return 0;
}
```

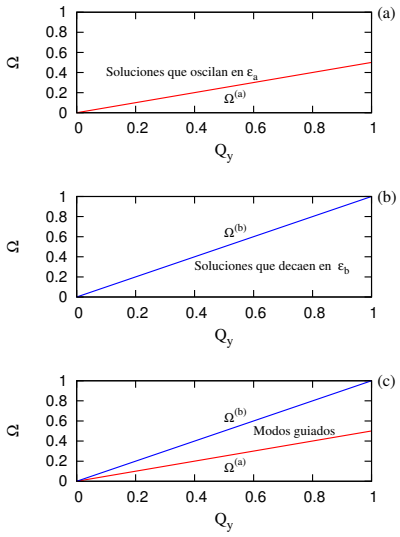

Figura A.1: Regiones - líneas de luz

Es conveniente tener una gráfica más elaborada para ilustrar la región de modos. La Figura A.1 se realizó con el programa "regiones.plt" el cual usa el archivo de datos "regiones.dat", los cuales se muestran a continuación.

A.4. Ecuación Trascendente Para Modos Pares

En esta sección se plantea la búsqueda de la solución para modos pares que se define mediante la ecuación

$$\tan(\pi Q_{ax}) = \frac{Q_{bx}}{Q_{ax}} \tag{A.8}$$

que también puede escribirse en la forma

$$f(Q_y, \Omega) = g(Q_y, \Omega) \tag{A.9}$$

donde

$$f(Q_y, \Omega) = \tan(\pi\sqrt{n_a^2\Omega - Q_y^2}) \tag{A.10}$$

y

$$g(Q_y, \Omega) = \frac{\sqrt{Q_y^2 - n_b^2\Omega}}{\sqrt{n_a^2\Omega^2 - Q_y^2}} \tag{A.11}$$

La ecuación A.8 se resuelve por medio del programa

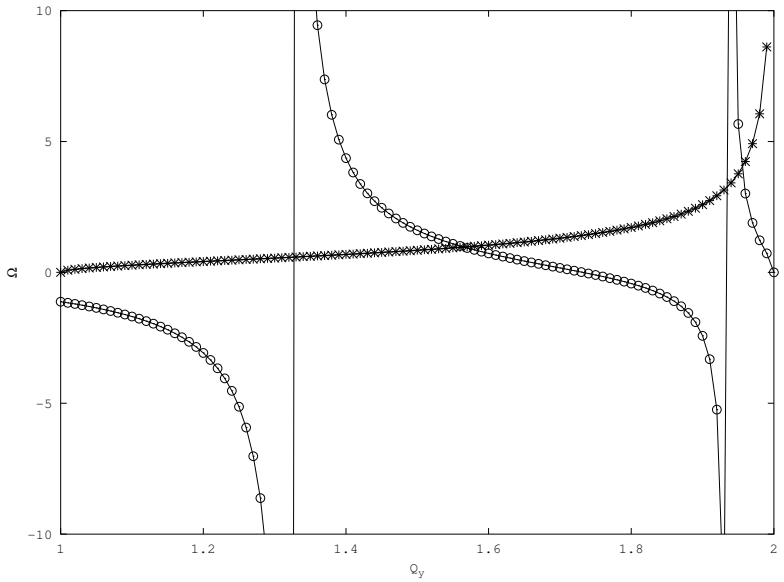

Figura A.2: Modos pares

Listado A.5: Modos pares

```
clear;

na = 1;
nb = 2.0;

Om = 1.0;
Qa = na * Om;
Qb = nb * Om;
Qy = [na * Om:0.01:nb * Om];

Qax = sqrt(Qy .* Qy - Qa * Qa);
Qbx = sqrt(Qb * Qb - Qy .* Qy);

f = tan(pi * Qbx);
g = Qax ./ Qbx;
grid on;
xlabel("Q_y");
ylabel("\\Omega");
plot(Qy, f, '-ob', Qy,g, '-*r')
axis([Qa Qb -10 10])
xlabel("Q_y");
ylabel("\\Omega");
print -deps 'par.eps'
```

A.5. Ecuación Trascendente Para Modos Impares

En esta sección se plantea la búsqueda de la solución para modos pares que se define mediante la ecuación

$$\cot(\pi Q_{ax}) = -\frac{Q_{bx}}{Q_{ax}} \tag{A.12}$$

que también puede escribirse en la forma

$$h(Q_y, \Omega) = i(Q_y, \Omega) \tag{A.13}$$

donde

$$hf(Q_y, \Omega) = \cot(\pi\sqrt{n_a^2\Omega - Q_y^2}) \tag{A.14}$$

y

$$i(Q_y, \Omega) = -\frac{\sqrt{Q_y^2 - n_b^2\Omega}}{\sqrt{n_a^2\Omega^2 - Q_y^2}} \tag{A.15}$$

La ecuación A.12 se resuelve por medio del programa

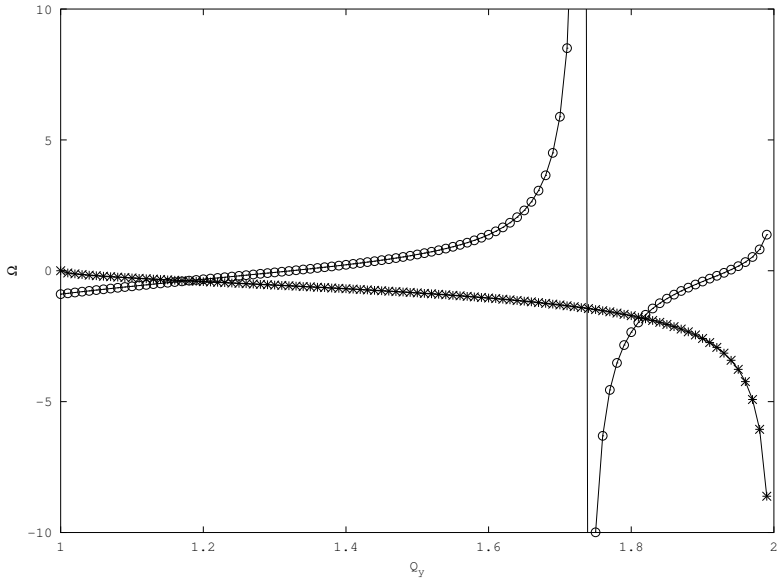

Figura A.3: Modos impares

Listado A.6: Modos impares

```
clear;

na = 1;
nb = 2.0;

Om = 1.0;
Qa = na * Om;
Qb = nb * Om;
```

```
Qy = [na * Om:0.01:nb * Om];

Qax = sqrt(Qy .* Qy - Qa * Qa);
Qbx = sqrt(Qb * Qb - Qy .* Qy);

f = cot(pi * Qbx);
g = -Qax ./ Qbx;
grid on;
xlabel("Q_y");
ylabel("\\Omega");
plot(Qy, f, '-ob', Qy, g, '-*r')
axis([Qa Qb -10 10])
xlabel("Q_y");
ylabel("\\Omega");
print -deps 'impar.eps'
```

Glosario

Air Objeto concreto que se deriva de la abstracción MaterialBase y que representa un material específico, en este caso aire.

Block1D Objeto concreto que se deriva de la abstracción GeometryBase y que representa una geometría de bloque en una dimensión.

Block2D Objeto concreto que se deriva de la abstracción GeometryBase y que representa una geometría de bloque en dos dimensiones.

CellConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para las celdas.

CellsBase Objeto base que sirve como abstracción para las celdas de un grid o malla.

Circle2D Objeto concreto que se deriva de la abstracción GeometryBase y que representa una geometría de círculo en dos dimensiones.

clase Término de programación orientada a objetos que se refiere a la definición de un objeto, es decir, la descripción del objeto.

ConfigBase Objeto base que sirve como abstracción para las configuraciones que se utilizan en el mecanismo de configuración.

ConfigJsonParser Objeto concreto que realiza las operaciones críticas en el mecanismo de configuración.

debian Sistema operativo o distribución totalmente libre basada en Linux.

DetectorBase Objeto base que sirve como abstracción para los detectores que son parte de una simulación.

DetectorConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para los detectores.

ElectroMagCells1D Objeto concreto que se deriva de la abstracción CellsBase y que representa un grupo de celdas en una dimensión que pertenecen a la malla de simulación electromagnética.

ElectroMagCells2D Objeto concreto que se deriva de la abstracción CellsBase y que representa un grupo de celdas en una dimensión que pertenecen a la malla de simulación electromagnética.

ElectroMagDetector1D Objeto concreto que se deriva de la abstracción DetectorBase y que representa un detector en una dimensión.

ElectroMagDetector2D Objeto concreto que se deriva de la abstracción DetectorBase y que representa un detector en dos dimensiones.

ElectroMagGaussian1D Objeto concreto que se deriva de la abstracción SourceBase y que representa una fuente gaussiana en una dimensión.

ElectroMagGaussian2D Objeto concreto que se deriva de la abstracción SourceBase y que representa una fuente gaussiana en dos dimensiones.

ElectroMagGaussianSin1D Objeto concreto que se deriva de la abstracción SourceBase y que representa una fuente gaussiana con componente sinusoidal en una dimensión.

ElectroMagGaussianSin2D Objeto concreto que se deriva de la abstracción SourceBase y que representa una fuente gaussiana con componente sinusoidal en dos dimensiones.

ElectroMagGrid1D Objeto concreto que se deriva de la abstracción GridBase y que representa una malla de simulación electromagnética en una dimensión.

ElectroMagGrid2D Objeto concreto que se deriva de la abstracción GridBase y que representa una malla de simulación electromagnética en dos dimensiónes.

ElectroMagnetic1D Objeto concreto que se deriva de la abstracción EngineBase y que representa un motor de cálculo en una dimensión.

ElectroMagnetic2D Objeto concreto que se deriva de la abstracción EngineBase y que representa un motor de cálculo en dos dimensiones.

ElectroMagPml1D Objeto concreto que se deriva de la abstracción PmlBase y que representa una capa de absorción en una dimensión.

ElectroMagPml2D Objeto concreto que se deriva de la abstracción PmlBase y que representa una capa de absorción en dos dimensiones.

ElectroMagSimulation1D Objeto concreto que se deriva de la abstracción SimulationBase y que representa una simulación electromagnética en una dimensión.

ElectroMagSimulation2D Objeto concreto que se deriva de la abstracción SimulationBase y que representa una simulación electromagnética en dos dimensiónes.

ElectroMagSinusoidal1D Objeto concreto que se deriva de la abstracción SourceBase y que representa una fuente sinusoidal en una dimensión.

ElectroMagSinusoidal2D Objeto concreto que se deriva de la abstracción SourceBase y que representa una fuente sinusoidal en dos dimensiones.

encapsulación Término de programación orientada a objetos que se refiere al hecho de que un objeto contenga toda la funcionalidad que brinda dentro de él mismo y solo exponga una interfaz pública de cómo utilizarla.

EngineBase Objeto base que sirve como abstracción para los engines o motores de cálculo que se utilizan en una simulación.

EngineConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para los motores de cálculo.

GeometryBase Objeto base que sirve como abstracción para las geometrías que son parte de una simulación.

GeometryConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para las geometrías.

Glass Objeto concreto que se deriva de la abstracción MaterialBase y que representa un material específico, en este caso vidrio.

GridBase Objeto base que sirve como abstracción para las mallas de simulación.

GridConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para la malla.

HDF5 Modelo de datos, librería y tipo de formato de archivo que sirve para almacenar y administrar datos.

herencia Término de programación orientada a objetos que se refiere a que un objeto puede heredar operaciones, propiedades y funcionalidad de un objeto *padre*, a éste se le llama objeto *hijo*.

MaterialBase Objeto base que sirve como abstracción para los materiales utilizados en una simulación o en una geometría que es parte de la simulación.

objeto Término de programación orientada a objetos que se refiere a una entidad en memoria que contiene operaciones así como propiedades y se utiliza para lograr una funcionalidad.

patron de diseño Un patrón define una posible solución correcta para un problema de diseño dentro de un contexto dado.

phoxonics Software especializado flexible, robusto y con módulos intercambiables desarrollado en la Universidad de Sonora a lo largo de la investigación doctoral que utiliza el Método de Diferencias Finitas en el Dominio del Tiempo para crear simulaciones del campo electromagnético en ambientes y medios configurables.

PixelGeometry2D Objeto concreto que se deriva de la abstracción GeometryBase y que representa una geometría en dos dimensiones construida a partir de una lista de puntos o pixeles.

PmlBase Objeto base que sirve como abstracción para la capa perfecta de absorción que es parte de una simulación.

PmlConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para la capa de absorción PML.

polimorfismo Término de programación orientada a objetos que se refiere a que de acuerdo al diseño del objeto éste se pueda comportar de distintas maneras según el contexto en el que fue creado.

SimulationBase Objeto base que sirve como abstracción para las simulaciones.

SimulationConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para la simulación.

SimulationManager Objeto concreto que realiza las operaciones sobre simulaciones, es decir, las administra.

SourceBase Objeto base que sirve como abstracción para las fuentes que son parte de una simulación.

SourceConfig Objeto concreto que se deriva de la abstracción ConfigBase y que representa la sección de configuración para las fuentes.

Vacuum Objeto concreto que se deriva de la abstracción MaterialBase y que representa un material específico, en este caso el vacio.

Water Objeto concreto que se deriva de la abstracción MaterialBase y que representa un material específico, en este caso agua.

Lista de Símbolos

Q_y Vector de Onda Reducido Para el Eje Y.

Ω Frecuencia Reducida.

λ Longitud de Onda.

μ Permeabilidad Magnética.

ω Frecuencia Angular.

$\rho(\mathbf{x}, t)$ Densidad de Carga.

θ Ángulo de Doblez.

ε Permitividad o Constante Dieléctrica.

$\mathbf{B}(\mathbf{x}, t)$ Inducción Magnética.

$\mathbf{D}(\mathbf{x}, t)$ Desplazamiento Eléctrico.

$\mathbf{E}(\mathbf{x}, t)$ Campo Eléctrico.

$\mathbf{H}(\mathbf{x}, t)$ Campo Magnético.

$\mathbf{J}(\mathbf{x}, t)$ Densidad de Corriente.

Lista de Abreviaturas

BPF Bandas Prohibidas Fotónicas.

CEI Circuitos Electrónicos Integrados.

CF Cristales Fotónicos.

CFI Circuitos Fotónicos Integrados.

CGS Centímetro-Gramo-Segundo.

CI Continuous Integration.

CPU Central Processing Unit.

CUDA Compute Unified Device Architecture.

FDTD Método de Diferencias Finitas en el Dominio del Tiempo.

FO Fibras Ópticas.

GPU Graphics Processor Unit.

JSON Javascript Object Notation.

MKS Metro-Kilogramo-Segundo.

MPI Message Passing Interface.

PML Perfectly Matched Layer.

TBB Threading Building Blocks.

TE Transversal Eléctrico.

TF Transformada de Fourier.

TM Transversal Magnético.

Mode conversion caused by bending in photonic subwavelength waveguides

Y. J. Rodriguez-Viveros[1], D. Moctezuma-Enriquez[2], P. Castro-Garay[1], B. Manzanares-Martinez[1], C. I. Ham-Rodriguez[3], E. Urrutia-Banuelos[3], and J. Manzanares-Martinez[3]

[1]Departamento de Fisica, Universidad de Sonora,
Blvd. Luis Encinas y Rosales, Hermosillo, Sonora 83000, Mexico.
yohan.jasdid@gmail.com, paola@cifus.uson.mx, betsabe.manzanares@correo.fisica.uson.mx

[2]Instituto Tecnologico de Hermosillo,
Av. Tecnológico s/n, Sonora 83170, Mexico
bersek_no1@hotmail.com

[3]Departamento de Investigacion en Fisica, Universidad de Sonora,
Blvd. Luis Encinas y Rosales, Hermosillo, Sonora 83000, Mexico
cavanharz@hotmail.com, eurrutia@cifus.uson.mx, jmanza@cifus.uson.mx

Abstract ─ We study the propagation of light in a subwavelength planar waveguide with an angular bend. We observe the mode conversion of a guided wave with a symmetric beam profile impinging into the bending of a waveguide. The guided wave outgoing from the bend is a mixed set of symmetric and asymmetric modes. The amount of mode conversion through the bend is quantified by calculating the Fourier Transform of the electric field profile. It is found that the conversion rate is a function of the bending angle.

Index Terms ─ Bending, mode conversion, waveguide, subwavelength.

I. INTRODUCTION

Optical Fibers (OF) are dielectric waveguides in the form of cylinders with a diameter (d) larger than the visible range wavelengths (λ=390 nm to λ=700 nm) [1]. OF are flexible waveguides, however, the angle of bending (θ) is limited by a critical angle (θ_c) defined by the principle of Total Internal Reflection (TIR) [1]. They are widely used in optical networks, sensors and other devices [2, 3]. Nowadays to control the flow of data on the Telecom Industry optical and electronic devices coexist. On the one hand, for long distances photons are transported via optical fibers, which support high bandwidths with low losses. On the other hand, in the electronic circuitry the essential functions of switching and routing are made by manipulating the flow of electrons. The bottleneck for data transfer in this hybrid technology is the operation frequency of the electronic processor which currently is of only a few GHz [4]. To increase the speed of data transfer in optical networks it has been proposed the development of Photonic Integrated Circuits (PIC) [5, 6]. A PIC does analogous main functions to those of Electronic Integrated Circuits (EIC). The difference is that PIC provides a direct manipulation of the photons traveling in optical waveguides. It is expected that the processing of light in PIC overcome the physical limitations of speed and power dissipation faced by EIC [7].

A progressive development of PIC has been made in the last two decades, which can be classified on three main approaches. A first approach is based on the use of Photonic Crystals (PC) as a framework for the design of photonic devices to control the flow of light [6]. PC are periodic dielectric structures composed by at least two materials in the unit cell of period (a) which is on the order of the visible wavelength, $a \sim \lambda$ [8]. The fundamental characteristic of PC is the existence of Photonic Band Gaps (PBG) where photons cannot propagate through the periodic lattice. Photonic devices on the order of visible wavelengths can be engineered in PC introducing localized states in the PBG by removing unit cells in the otherwise perfect crystal [9]. A second strategy to design PIC is based on the use of plasmons [10]. Plasmon-based approaches require a precise manipulation of the electromagnetic fields, which are exponentially decaying from the surfaces [11]. It has been suggested the possibility of creating surface-plasmon circuitry with photonic components smaller than the diffraction limit of light [12]. Plasmons however have substantial optical losses which can severely limit their applicability [13]. To overcome these losses, it has recently been proposed the use of gain-loss structures which introduce unusual possibilities for the propagation of plasmons [14]. A third approach to fabricate PIC is based on the use of

AIP | Applied Physics Letters

Non-perpendicular hypersonic and optical stop-bands in porous silicon multilayers

J. Manzanares-Martinez, D. Moctezuma-Enriquez, Y. J. Rodriguez-Viveros, B. Manzanares-Martinez, and P. Castro-Garay

Citation: Appl. Phys. Lett. **101**, 261902 (2012); doi: 10.1063/1.4773243
View online: http://dx.doi.org/10.1063/1.4773243
View Table of Contents: http://apl.aip.org/resource/1/APPLAB/v101/i26
Published by the American Institute of Physics.

Non-perpendicular hypersonic and optical stop-bands in porous silicon multilayers

J. Manzanares-Martinez,[1,a)] D. Moctezuma-Enriquez,[2,b)] Y. J. Rodriguez-Viveros,[3] B. Manzanares-Martinez,[3] and P. Castro-Garay[3]
[1]*Departamento de Investigacion en Fisica, Universidad de Sonora, Apartado Postal 5-088, Hermosillo, Sonora 83180, Mexico*
[2]*Centro de Investigacion en Materiales Avanzados (CIMAV), Miguel de Cervantes 120, Chihuahua 31109, Mexico*
[3]*Departamento de Fisica, Universidad de Sonora, Blvd. Luis Encinas y Rosales, Hermosillo, Sonora 83180, Mexico*

(Received 21 September 2012; accepted 10 December 2012; published online 26 December 2012)

We study by theoretical simulations the non-perpendicular propagation of electromagnetic and elastic waves in porous silicon multilayers. We proceeded in three steps. First, we found the conditions to obtain a simultaneous photonic-phononic mirror at normal incidence. Second, we determined the angular variation of the mirrors computing the projected band structure. In a third step, we found out, on the one hand, that there are no conditions to obtain an omnidirectional mirror for electromagnetic waves. But, on the other hand, we found the conditions were possible to obtain an omnidirectional mirror for elastic waves. Moreover, the elastic mirror is revealed to be a polarization-converter due to the conversion of evanescent modes in the band gap. © *2012 American Institute of Physics*. [http://dx.doi.org/10.1063/1.4773243]

It is well known that the propagation of electromagnetic and elastic waves in periodic structures can be forbidden for certain ranges of frequencies, giving rise to the photonic and phononic band gaps (PtBG and PnBG, respectively).[1,2] For many years, the fields of photonic crystals (PtC) and phononic crystals (PnC) have remained separate and have evolved with similar methods of treatment, but considering lattices of different scale. A typical lattice period for a PtC with PtBG in the visible range is $d_{PtC} = 290\,\text{nm}$,[3] while the lattice period for a PnC in the audible regime is $d_{PnC} = 1\,\text{cm}$.[4] The PnC period is much bigger than the PtC period, $d_{PnC} \gg d_{PtC}$.

In 2006, Maldovan and Thomas explored the simultaneous localization of photons and phonons in two-dimensional (2D) periodic structures.[5] As a prerequisite for the existence of such localization of light and sound, it was found necessary to have a periodic dielectric/elastic structure possessing both PtBG and PnBG. They found that a periodic structure made of air holes in a solid block of silicon exhibits large and simultaneous PtBG and PnBG for all directions of the periodicity plane. They found that if the periodic structure is fabricated with a lattice period of $d = 150\,\text{nm}$, there is a PtBG for transversal electric (TE)-modes (no electric field in the direction of propagation) between 4.64×10^{14} and $5.80 \times 10^{14}\,\text{Hz}$. A complete PnBG for hypersonic frequencies between 1.36×10^{10} and $2.16 \times 10^{10}\,\text{Hz}$ was also determined. It is important to note that the range of frequencies for electromagnetic and elastic band gaps is different because the forbidden wavelength for light (λ_{light}) and sound (λ_{sound}) is of the order of the lattice period ($\lambda_{light} \sim \lambda_{sound} \sim d$). The PtBG and PnBG do not need to overlap in frequencies to consider the simultaneous control of light and sound by the same structure.

Later, other works studied the photonic and phononic band structures of more complicated dielectric-elastic periodic lattices.[6–10] The existence of cavities, where the simultaneous localization of light and sound permits to enhance the interaction of photons and phonons, has been explored for 1D[11] and 2D structures,[5,12] respectively. To define a crystalline structure that supports both PtBG and PnBG, in 2009, Sadat-Saleh *et al.* introduced the notion of phoxonic crystals (PxCs).[8] PxCs offer the potential to engineer light and sound allowing the molding of flow acousto-optics waves within these structures.

Recently, it has been reported that hypersonic phononic crystals (HPnCs) have been designed and fabricated to control phonons of high energy frequency between 1 GHz and 100 GHz.[13] HPnCs open a pathway to explore new physical phenomena. For example, an exciting possibility is to manage the thermal flow. In effect, a PnBG perturbs the phononic density of states, which impacts in physical quantities such as the thermal conductivity and heat capacity.[14] Since the lattice period of HPnC is comparable with the wavelength of light, it could also be possible—in principle—to manage the phononic density of states with light using acousto-optical interactions. This opens the possibility to create a new generation of acousto-optical devices.[15]

Porous silicon (PSi) is a flexible material used to fabricate structures with 1D dielectric-elastic periodic impedance. The control of the porosity allows to change the optical and elastic properties. On the one hand, the difference of porosity allows to vary the refractive index. On the other hand, the change of porosity varies the density of the material and the longitudinal and transverse sound speed. The existence of optical Bragg mirrors characterized by band gaps in the

[a)]Also at Centro Mesoamericano de Fisica Teorica (Mesoamerican Centre for Theoretical Physics MCTP), Ciudad Universitaria, Carretera Emiliano Zapata Km-4, Tuxtla Gutierrez, Chiapas, Mexico.
[b)]Also at Departamento de Fisica, Universidad de Sonora, Blvd. Luis Encinas y Rosales, Hermosillo, Sonora 83180, Mexico. Electronic address: papers@phoxonics.com.

